Carlos Pillajo / Roberto Hincapié

Wireless Network Control Systems

De la teoría a la práctica

Universidad Politécnica Salesiana

Wireless Network Control Systems

De la teoría a la práctica

Carlos Pillajo
Roberto Hincapie

Wireless Network Control Systems

De la teoría a la práctica

ABYA YALA | UNIVERSIDAD POLITÉCNICA SALESIANA

2018

WIRELESS NETWORK CONTROL SYSTEMS
De la teoría a la práctica
© *Carlos Pillajo y Roberto Hincapie*

1ra edición: Universidad Politécnica Salesiana
Av. Turuhuayco 3-69 y Calle Vieja
Cuenca-Ecuador
Casilla: 2074
P.B.X. (+593 7) 2050000
Fax: (+593 7) 4 088958
e-mail: rpublicas@ups.edu.ec
www.ups.edu.ec

CARRERA DE INGENIERÍA ELECTRÓNICA

Derechos de autor: 054864

ISBN: 978-9978-10-326-5

Edición, diseño, diagramación e impresión: Editorial Universitaria Abya-Yala
Quito-Ecuador

Tiraje: 300 ejemplares

Impreso en Quito-Ecuador, octubre de 2018

Publicación arbitrada de la Universidad Politécnica Salesiana

Índice

Índice 5
Índice de figuras 9
Índice de tablas 14
Agradecimientos 15
Introducción 17

Capítulo uno
Un perfil sobre los sistemas de control en red inalámbrico (WNCS)

Antecedentes 22
Los WNCS 24
Visión de los sistemas de control en red 25
Control a través de la red de comunicación 26
Control basado en eventos 29
Introducción al control basado en eventos 30
Características de los WNCS 35
Sistemas inalámbricos: las imperfecciones de la red inalámbrica 36
Modelo de retardo 39
Modelo de abandono de paquetes 42

Capítulo dos
Controlador basado en eventos

Sistemas basados en eventos 43
Muestreo basado en eventos 45
Sistemas de datos muestreados 45
Las técnicas de muestreo y control basadas en eventos 47
Estrategias de control basado en eventos 51

El controlador PID basado en eventos 54
Alternativas para el controlador PID basado en eventos 58
El objetivo del control basado en eventos 60
Realimentación de estado continuo 63
Realimentación de estados basado en eventos 65
El uso de predictores en el control basado en eventos 70
Sintonización de los controladores PID 72
Clásica sintonización del PID 73
Método de Ziegler-Nichols 74
Asignación de polos 75
Los algoritmos genéticos para la sintonización PID 77
Algoritmo heurístico para la sintonización del PID basado en eventos en un WNCS 78
Optimización mediante el cúmulo de partículas (PSO) 78
Algoritmo PSO para sintonizar un controlador PID 83
Optimización mediante algoritmo genético (AG) 85

Capítulo tres
Implementación de un WNCS aplicado al controlador PID predictivo basado en eventos para el sistema BB

Descripción de la arquitectura de los dispositivos que conforman el WNCS 97
Elementos de red 99
Servidor-controlador 100
Cliente sensor-actuador 101
El modelo del sistema Beam-Ball 105
Metodología para la simulación e implementación 108
Diseño del controlador PID predictivo para la planta BB 112
Algoritmo predictivo para el control PID basado en eventos para el sistema BB 113
Definición de los requerimientos 113
Identificación de los módulos del algoritmo implementado 115
Algoritmos implementados en el WNCS 116
Algoritmo en el servidor-controlador predictivo PID asíncrono 117
Algoritmo predictivo con PID síncrono y semi-síncrono 118
Algoritmos en el cliente sensor-actuador 120
Definición de la mejor condición para el evento del sistema 129

Capítulo cuatro
Implementación de un WNCS local y remoto para el sistema BB mediante Raspberry Pi y Arduino

Descripción de los dispositivos del sistema 132
Servidor Raspberry Pi-3 132
El cliente 132
Herramientas para implementar algoritmos a través de la nube 133
PHP (Hypertext Preprocessor) 133
Base de datos MySQL 134
JSON (notación de objetos de JavaScript) 134
Diseño de la implementación del WNCS 134
Arquitectura de la planta 136
Esquema de funcionamiento local 141
Esquema de funcionamiento en la nube 142
Procedimiento para la implementación del sistema 143
Esquema en PHP y MYSQL 143
Algoritmos implementados 146
Ensayos del sistema WNCS 156

Capítulo cinco
Implementación de un WNCS para el control de voltaje de un generador DC didáctico

Descripción de la arquitectura de los dispositivos que conforman el WNCS 159
Elementos de red 161
Servidor-controlador 162
Cliente sensor-actuador 165
El modelo del sistema del control de voltaje de un motor-generador DC 168
Metodología para la simulación e implementación 169
Diseño del controlador PID predictivo para la planta 170
Algoritmo predictivo para el control PID basado en eventos para el sistema 171
Algoritmos implementados en el WNCS 172
Algoritmo en el servidor-controlador predictivo PID asíncrono 172
Algoritmos en el cliente sensor-actuador 173
Resultados de la aplicación 175

Anexos

Anexo A. Modelo matemático de la planta BB 179
Anexo B. Modelo matemático del sistema motor-generador..... 192

Bibliografía 203

Índice de figuras

Figura 1. Esquema de un sistema de control basado en red inalámbrica 17

Figura 2. Estructura de una WNCS (izquierda) y retardos en la comunicación de datos (derecha) 27

Figura 3. Lazo de control basado en eventos 31

Figura 4. Transmisión inalámbrica irregular..... 36

Figura 5. Estructura directa de la WNCS 39

Figura 6. Diagrama de retardos causados en la red 41

Figura 7. Modelo de pérdida de paquetes Gilbert-Elliot 42

Figura 8. Esquema de un sistema de control basado en eventos 45

Figura 9. Enfoques de sistemas de datos muestreados 46

Figura 10. Muestreo basado en eventos..... 50

Figura 11. Diagrama de bloques simplificado de un WNCS 52

Figura 12. Diagrama de bloques y representación simplificada de un controlador PID 54

Figura 13. Intercambio de información en el enfoque PID basado en eventos 57

Figura 14. Generador de la entrada de control 66

Figura 15. Lazo de control de realimentación de estado basado en eventos..... 69

Figura 16. Diagrama de flujo para el algoritmo PSO 82

Figura 17. Algoritmo genético..... 86

Figura 18. Diagrama de flujo del algoritmo genético 88

Figura 19. Respuestas del sistema a una señal paso 92
Figura 20. Error generado por el algoritmo genético bajo el criterio IAE 93
Figura 21. Constantes K_p, K_i, K_d con AG 93
Figura 22. Error generado por el algoritmo PSO bajo el criterio IAE 94
Figura 23. Constantes K_p, K_i, K_d con algoritmo PSO 95
Figura 24. Elementos de red 97
Figura 25. Diagrama esquemático del módulo servidor 98
Figura 26. Diagrama esquemático del módulo cliente 98
Figura 27. Configuración del *router* Tp-Link 99
Figura 28. Configuración de red WAN 99
Figura 29. Configuración del protocolo DHCP 100
Figura 30. Parámetros de red en Arduino 101
Figura 31. Red Wi-Fi de Dragino Yun 102
Figura 32. Configuración de Dragino Yun 103
Figura 33. Librerías de Dragino Yun 103
Figura 34. Dragnino Yun como cliente 104
Figura 35. Comunicación cliente-servidor 104
Figura 36. Esquema del sistema BB 106
Figura 37. Diagrama de bloques para el control de BB 109
Figura 38. Diagrama de control en cascada simulado 111
Figura 39. Sistema BB en su representación Z 112
Figura 40. Muestreo asíncrono con información adicional durante la ausencia de comunicación 114
Figura 41. Dirección de la pelota por inclinación del riel 114
Figura 42. PID asíncrono 1 117

Figura 43. PID asíncrono 2 118
Figura 44. Tipos de muestreo 120
Figura 45. PID asíncrono con intervalo de trabajo temporizado 122
Figura 46. Retardo del PID asíncrono con intervalo de trabajo temporizado 122
Figura 47. PID asíncrono con variación de variable medida 124
Figura 48. Retardos del PID asíncrono con variación de la variable medida 124
Figura 49. PID asíncrono con error integral 126
Figura 50. Retardos del PID asíncrono con error integral 126
Figura 51. PID Asíncrono con error integral al cuadrado 128
Figura 52. Datos de los tiempos de retardo en PID asíncrono con error integral al cuadrado 128
Figura 53. Esquema didáctico del proyecto 135
Figura 54. Esquema de control local en Raspberry Pi-3 135
Figura 55. Esquema de control en la nube 136
Figura 56. Estructura de Arduino y Dragino Yun 137
Figura 57. Interfaz principal 139
Figura 58. Flujograma del controlador 140
Figura 59. Esquema del controlador local 141
Figura 60. Esquema de controlador en la nube 142
Figura 61. Iteración de lenguajes para el monitoreo 144
Figura 62. Linealizar la entrada analógica 148
Figura 63. Conexión con el servidor 148
Figura 64. Algoritmo predictivo 149
Figura 65. Muestra de resultados de la planta 149
Figura 66. Estado de desconexión del sistema 151

Figura 67. Archivos relacionados al proyecto 153
Figura 68. Base de datos MySql forma local..... 154
Figura 69. Sentencias MySql utilizadas..... 155
Figura 70. Dispositivos utilizados en la implementación del WNCS..... 159
Figura 71. Dispositivos asociados al servidor-controlador 160
Figura 72. Diagrama esquemático del módulo cliente (sensor-actuador)..... 160
Figura 73. Diagrama esquemático del sistema WNCS 161
Figura 74. Dirección IP Raspberry 162
Figura 75. Dirección IP Raspberry 163
Figura 76. Carpeta del servidor que contiene el controlador..... 164
Figura 77. Decodificación de los datos formato JSON (servidor)..... 164
Figura 78. Codificación de los datos formato JSON (servidor)..... 165
Figura 79. Librerías utilizadas en el cliente 165
Figura 80. Dirección IP del servidor local..... 166
Figura 81. Decodificación de datos JSON (cliente) 166
Figura 82. Codificación de datos JSON (cliente)..... 167
Figura 83. Diagrama de comunicación cliente-servidor 168
Figura 84. Diagrama del sistema controlador de voltaje DC..... 169
Figura 85. Gráfica de la muestra voltaje de salida vs señal PWM..... 169
Figura 86. Control PID en el servidor..... 171
Figura 87. Algoritmo implementado en el servidor 173
Figura 88. Algoritmo implementado en la planta..... 174
Figura 89. Control PID predictivo en el cliente..... 175
Figura 90. Algoritmo implementado en el controlador 1 176
Figura 91. Algoritmo implementado en el controlador 2 176

Figura 92. Herramienta IDENT para sacar FT 180
Figura 93. Gráfico de polos y ceros de la FT sin controlador 183
Figura 94. Repuesta paso de la planta sin controlador..... 184
Figura 95. Respuesta a una señal paso del sistema con PID..... 185
Figura 96. LGR de la planta con controlador PID continúo..... 185
Figura 97. Control proporcional de posición para la riel..... 186
Figura 98. Diagrama de la función de transferencia de un motor DC..... 187
Figura 99. Diagrama del modelo de la bola con retenedor de orden cero 189
Figura 100. Diagrama de bloques en Simulink del sistema BB..... 191
Figura 101. Respuesta del sistema BB ante una entrada paso en Simulik..... 191
Figura 102. Esquema motor DC 192
Figura 103. Diagrama de torques en el rotor 194
Figura 104. Ventana IDENT 197
Figura 105. Ventana Import Data 198
Figura 106. Número de polos y ceros del sistema..... 199
Figura 107. Estimación de función de transferencia 199
Figura 108. Modelamiento del sistema (primera muestra)..... 200
Figura 109. Modelamiento del sistema (segunda muestra) 200
Figura 110. Resultados presentados por Matlab 201

Índice de tablas

Tabla 1. Condición lógica utilizada para el muestreo por eventos 50
Tabla 2. Datos de los tiempos de retardo en PID asíncrono con criterio IAE 96
Tabla 3. Algoritmo nodo servidor-controlador..... 116
Tabla 4. Datos de los tiempos de retardo en el PID asíncrono con intervalo de trabajo temporizado..... 123
Tabla 5. Datos de los tiempos de retardo en PID asíncrono con variación de la variable medida..... 125
Tabla 6. Datos del tiempo de retardo en PID asíncrono con acumulación del error..... 127
Tabla 7. Datos del tiempo de retardo en PID asíncrono con error integral al cuadrado..... 129
Tabla 8. Datos de las condiciones para eventos..... 130
Tabla 9. Tabla comparativa del controlador local/controlador en la nube 157

Agradecimientos

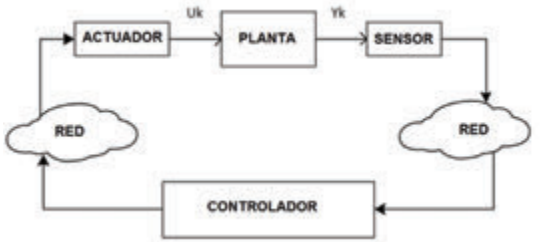
El conocimiento no es de quien lo tiene… sino de quien lo aplica.

Expreso mi gratitud a todas las personas que han hecho la realidad este texto, a mi mentor PhD Roberto Hincapié, gracias por sus aportes, sugerencias y motivaciones, este texto es producto de la investigación teórica, analítica de proyectos de ingenieros de la Universidad Politécnica Salesiana que aportaron con sus aplicaciones empíricas que fueron hechas realidad en base a la investigación aplicada ellos son: Agustín Bonilla, Diego Soberon, Adrián López y Diego Zapata.

Introducción

El diseño de Sistemas Cyberfísicos (CPS) en general y en particular del Wireless Network Control Systems (WNCS), requiere una atención simultánea de los aspectos de comunicación, computación y de control, además de los fenómenos físicos. Cuando se utiliza la comunicación inalámbrica, una atención especial debe darse al recurso crítico de control de la comunicación, ya que es a la vez limitado y compartido entre otros dispositivos. Actualmente, la tecnología inalámbrica es utilizada en muchas áreas como la electrónica de consumo, redes de sensores, automatización en la industria automotriz, etc. Los WNCS (figura 1) son un tipo de los NCS, que son sistemas de control distribuidos espacialmente, donde la comunicación entre sensores, controlador y actuador es compartida en una red inalámbrica, en donde los WNCS tienen limitaciones como el ancho de banda, pero también muchas ventajas como arquitectura flexible, bajo costo de mantenimiento, disminución del cableado y alta movilidad (Haupt en Lunze, 2014). Además, dan la posibilidad de recurrir a la intranet de una empresa o a internet, para comunicar los diferentes componentes del sistema de control, pudiendo estar el controlador lejos del proceso y cerrando el lazo de control a través de internet.

Figura 1
Esquema de un sistema de control basado en red inalámbrica



Fuente: los autores

Tradicionalmente, todas las teorías de control moderno y clásico se centran en el estudio de sistemas dinámicos interconectados a través de canales de comunicación ideales. Mientras que la teoría de comunicaciones estudia la transmisión de información a través de canales imperfectos, en los que se pueden producir retardos y pérdidas de información. Por tanto, un WNCS estaría en la intersección de ambas teorías, como menciona Hespanha (*et al.*, 2007).

Los sistemas de control en red, por su propia naturaleza, presentan varios problemas:

- Retardos variables: debido a que la información en sí es un conjunto de paquetes que viaja por las redes y tardan tiempo en transmitirse a través de la red, pues esta información depende de condiciones muy variables como la congestión de la red o la calidad del canal de comunicación; también el tiempo de retardo se debe al tiempo que demora en muestrear la señal continua de la planta en el lado del cliente y el procesamiento de datos en el lado del controlador.
- Desconexiones: dependiendo de la calidad de servicio de la red, sobre todo en redes inalámbricas, son frecuentes las desconexiones momentáneas que pueden ocasionar pérdidas de información y que, por un momento, el sistema de control quede sin servicio.
- Ancho de banda limitado: esto debido a que actualmente existen muchos dispositivos que pueden hacer uso de la red, limitando la capacidad de transmisión.

Los problemas anteriormente citados se pueden agravar en el caso de procesos que presentan dinámicas inestables y rápidas, debido a que los procesos requieren tiempos de actuación menores que la sumatoria de retardos debido a las comunicaciones. El uso de redes inalámbricas para sensores en aplicaciones de monitoreo se ha masificado, no así para los sistemas de control retroalimentado en tiempo real, pues su uso aún es incipiente. Las aplicaciones de supervisión no necesitan grandes requisitos de tiempo real y no sufren de problemas críticos re-

lacionados con retardos en el tiempo, a comparación de los problemas típicos de control de retroalimentación. Para superar estos problemas de tiempo hay dos enfoques: la primera opción es optimizar la infraestructura de comunicación para un controlador dado, es decir, realizar una comunicación dedicada o adquirir el hardware más costoso disponible para alcanzar una comunicación tan cerca a la ideal como sea posible, que todavía puede resultar en un comportamiento insatisfactorio debido a las limitaciones del hardware; la segunda opción es optimizar el controlador para una infraestructura de comunicación dada, es decir, desarrollar y aplicar algoritmos de control que funcionen y que estén basados en la teoría de WNCS, para de esta manera garantizar un comportamiento deseable en lazo cerrado.

En este libro exploramos la segunda opción, cuyo objetivo es proporcionar las bases para el desarrollo e implementación de un controlador basado en eventos con algoritmo predictivo en un sistema de red inalámbrico, es decir, un WNCS. En un sentido más genérico, el firmware que puede ser desarrollado e implementado en una tarjeta Arduino o Raspberry Pi-3, constituye el controlador en el cual se realiza el algoritmo PID predictivo basado en eventos, donde los datos de control los transmite al sensor-actuador.

Tal como se menciona en Åström (*et al.*, 1989), la retroalimentación es un concepto que por sí mismo ha revolucionado la manera en cómo se realiza el control, en ese mismo artículo también se señala que el controlador PID es la forma más utilizada de control por retroalimentación. Un aspecto importante que se debe considerar del controlador PID es la estructura del mismo en la cual se procesa el error; se sabe que todo control PID debe ser capaz de manejar las ganancias proporcional, derivativa e integral, sin embargo, existen diversas maneras de hacerlo, es decir, existen diversas estructuras de PID. Para este proyecto se plantea la implementación basada en eventos, pero sin importar cuál estructura se haya elegido para el controlador, se debe considerar que este deber ser sintonizado, es decir, se deben elegir valores para cada una

de las ganancias del controlador, esto es lo que se conoce como sintonización. Este proceso de sintonización es necesario que se realice con cada nueva planta o cuando el desempeño no es satisfactorio. Esta es una característica del controlador que lo hace tan versátil.

En este texto se implementan dos técnicas de sintonización basadas en heurísticas, que resuelve el problema de la sintonización de un sistema Beam-Ball, el cual es un proceso que requiere realimentación para su estabilización. Los algoritmos genéticos y los algoritmos PSO son técnicas heurísticas, las cuales buscan encontrar una solución, partiendo de un espacio de búsqueda específico que se define en un principio y que va evolucionando conforme el algoritmo avanza. Ya se han reportado artículos (Santos, 2014; Yajuan y Qinghai, 2011) donde se utiliza un algoritmo basado en heurísticas para la sintonización de las ganancias de controladores de tipo PID. Los resultados obtenidos en la mayoría de las ocasiones han sido satisfactorios o al menos prometedores. Es importante remarcar que a pesar de que los resultados son satisfactorios, la mayoría de ellos provienen de simulaciones de los modelos matemáticos del proceso.

En el capítulo uno del presente trabajo se realizará una breve explicación sobre los antecedentes y características de los WNCS. En el capítulo dos revisaremos los conceptos y características de los sistemas de control en red inalámbricos basados en eventos, para fundamentar la base teórica sobre la cual se presentará una alternativa de controlador PID. En el tercer capítulo se estudiará el controlador PID basado en eventos con estimadores para sistemas lineales y algoritmos de sintonización heurística. Finalmente, en el cuarto y quinto capítulos se realizarán aplicaciones en donde se determinará el modelo de la planta sobre el cual se va a implementar un controlador PID predictivo y se implementará el controlador PID inalámbrico para dos plantas.

Capítulo uno
Un perfil sobre los sistemas de control en red inalámbrico (WNCS)

En la actualidad muchos paradigmas se están desvaneciendo, sobre todo en los sistemas de control, gracias a los CPS, que son aquellos que involucran conocimientos en el área de control, comunicación y computación. Muchos de los retos actuales en el contexto del control de retroalimentación en tiempo real requieren que los canales de comunicación entre sensores, controladores y actuadores deban ser compartidos con otras aplicaciones. Se sabe que el compartir la información a través de la red tiene relación con propiedades que son fundamentales a los sistemas de control retroalimentado en tiempo real, tales como la velocidad de datos, los retardos y el uso del canal de comunicaciones, propiedades propias de la red que en su mayoría son inciertas. Los sistemas de control actuales tratan de optimizar los recursos de comunicación debido a que las incertidumbres, dadas sobre todo por las redes de comunicación inalámbricas, hacen replantear los supuestos básicos en la implementación de los controladores en lazo cerrado de realimentación, por lo tanto, el campo de los WNCS actualmente está en desarrollo y más aún sus implementaciones. Debido a esto se están generando algoritmos que permitan garantizar propiedades de control como la estabilidad y el rendimiento, en presencia de la comunicación incierta.

Se debe tomar en cuenta la ocupación de los canales de comunicación inalámbrica, más aún cuando la distancia es grande y se va a utilizar para acciones de control remoto en donde también influye el escalamiento de los procesos. Los sistemas de control que se extienden a través de grandes distancias, a menudo se desea controlarlos de manera

descentralizada. Por lo tanto, la combinación de la teoría de sistemas de control en red (NCS) y la teoría de comunicación inalámbrica para el control, abre un campo de investigación muy grande en temas sobre los que trata este proyecto. Antes de profundizar en la contribución específica de este proyecto, se dará una breve visión general de cómo los avances en telecomunicaciones ya han permitido algunas aplicaciones pertinentes a los sistemas de control, ya que para nuestro caso específico se plantea desarrollar algoritmos de control para dos sistemas: Beam-Ball y un motor-generador DC.

Antecedentes

Con la utilización a gran escala de los teléfonos inteligentes se ha provocado que la demanda de datos se dispare, esto ha producido que el campo de las telecomunicaciones crezca a pasos agigantados en un esfuerzo para satisfacer esta demanda. Hoy en día se puede acceder a una conexión de internet cada vez más rápida en casi cualquier lugar del mundo mediante un teléfono inteligente, mientras que hace solo en veinte años para tener acceso a internet se necesitaba un gran computador con un módem de acceso telefónico de conexión (56 Kb/s). Estas redes de alta velocidad se pueden conectar fácilmente y pueden ser utilizadas en una gran variedad de aplicaciones relacionadas con el control, tales como redes inalámbricas de sensores (WSN) y control de movimiento.

Las WSN ya han tenido éxito en el uso de telecomunicación inalámbrica. Estas redes se basan en dispositivos de bajo costo y consumo (nodos) que son capaces de obtener información de su entorno, procesarla localmente y comunicarla a través de enlaces inalámbricos hasta un nodo central de coordinación, como lo describe ampliamente Fernández Barcell (2008). Estos avances se han implementado en una gran variedad de aplicaciones: sensado industrial (Chong *et al.*, 2003; Dunbar, 2001), aplicaciones para la salud (Flores Carbajal, 2012) y proyectos agrarios (García, 2014), por mencionar solo algunos.

En el pasado, los avances de telecomunicaciones fueron capaces de proporcionar una alta calidad de servicio para aplicaciones de control (por ejemplo, a través de un bus CAN dedicada). Actualmente, los avances en telecomunicaciones más recientes, como redes inalámbricas e internet de alta velocidad, imponen la necesidad de desarrollar la teoría de control para utilizar las comunicaciones de forma fiable en un contexto de sistemas de control en tiempo real.

Las WSN han demostrado funcionar satisfactoriamente en aplicaciones de monitoreo y supervisión, lo cual se puede atribuir a su dinámica; mientras tanto, los datos relacionados como las aplicaciones de control en tiempo real aún no han despegado en sus aplicaciones. Es por esto que se debe primero tener mayor comprensión de la teoría de las WNCS, la cual está relacionada con la comprensión de propiedades de control como estabilidad y rendimiento, en correlación con las propiedades del medio incierto de comunicación.

Una de las importantes ventajas de las redes de información actuales es la capacidad de comunicarse a través de grandes distancias sin cables dedicados, ya que la distribución física de estos a través de cientos de metros puede volver a una aplicación en particular muy costosa y poco flexible. Asimismo, la aplicación de un controlador centralizado que deba recoger los datos relacionados con el control con el fin de regular los sistemas que se extienden a través de cientos de metros, es muy poco práctica debido a las distancias largas de cables de comunicación. El uso de las redes de comunicación actuales ofrece una opción para la aplicación de sistemas de control ya no basados en la instalación de cables dedicados entre todos los dispositivos de control.

Así, la combinación de la teoría WNCS y la teoría de control descentralizada surge como un campo de investigación de gran relevancia. Es por eso que en este trabajo se utiliza la teoría de WNCS para la implementación un controlador inalámbrico basado en eventos, pues aunque los ingenieros de control tienen mayor confianza en las comunicaciones punto a punto cableadas, mediante este texto se pretende dar una op-

ción funcional para que se abra paso a proyectos de sistemas dinámicos de control en donde las comunicaciones del sensor al controlador y del controlador al actuador no sean cableadas, es decir, que se pueda liberar la dependencia del cableado dedicado para la comunicación, resolviendo de esta manera muchos desafíos actuales y abriendo muchas posibilidades para probar teorías que están surgiendo y que constituye un campo de investigación muy relevante.

Los WNCS

Los sistemas dinámicos de control necesitan comunicar información emitida por el sensor hacia el controlador y la información del controlador hacia el actuador. Los sistemas de control a través de redes digitales han sido implementados desde hace mucho tiempo, pero el creciente interés en el campo de los sistemas de control en red se debe a que en la actualidad existe una gran variedad de redes digitales disponibles en todas partes, en las cuales se puede implementar el lazo de realimentación sin costo adicional. Las conexiones inalámbricas facilitan la extensión del área de aplicación de los sistemas de control automático hacia objetos móviles, pues las medidas de las variables y el control de las variables pueden ahora ser transmitidos al controlador desde casi todos los lugares de una planta.

Una definición del concepto WNCS sería: *un sistema de control en lazo cerrado que tienen que tomar en consideración los sistemas en red inalámbrico para la transmisión de datos*, esta definición comprende que las redes de datos imponen desafíos que no pueden ser considerados de forma independiente del sistema de control de realimentación. Esto revela un cambio en la percepción de las redes de control de retroalimentación con WNCS, convirtiéndose en una de las áreas de investigación más activas de hoy en día. En la siguiente sección, se describirá brevemente la aparición de NCS en un nivel general.

Visión de los sistemas de control en red

Los sistemas de control realimentado han sido beneficiados de las redes digitales, pues debido a estos los sistemas de comunicación se han descentralizado y distribuido sus acciones de control, pero en la actualidad los requisitos en tiempo real de aplicaciones industriales necesitan de redes altamente especializadas, razón por lo cual desde hace varios años se han desarrollado buses de campo. Uno de los primeros buses de campo en el mercado fue CAN (Controller Area Network), luego se introdujeron en el mercado buses de campo industriales como: Interbus y Device-Net, Profibus y Foundation Fieldbus. Por tanto, nos referimos a este tipo de redes como las "redes deterministas", las cuales proporcionan soluciones de comunicación patentadas para aplicaciones de control y son estándar en el control industrial en la actualidad.

Alrededor del año 2000, los esfuerzos se dirigieron a utilizar las redes con un comportamiento no determinístico para fines de control. El gran avance se debió a desarrollos tecnológicos como las mejoras de las tecnologías de comunicación inalámbrica, por un lado, y a los avances en las redes de computadoras en general (en particular, Ethernet), por otra parte, avances que también allanaron el camino para la expansión de internet. Por tanto, se han realizado grandes esfuerzos para emplear estas redes en el control realimentado. La ventaja de las redes de ordenadores en general, tales como internet, es que la infraestructura de comunicación ya existente puede ser utilizada sin costes de instalación adicionales, en contraste con los buses de campo. Estas redes (WNCS) también son no propietarias, por lo que no hay derechos de licencia que deban ser pagados.

Las redes inalámbricas con tecnologías como: Bluetooth (IEEE 802.15.1), Wi-Fi (IEEE 802.11) y ZigBee (IEEE 802.15.4) han experimentado un rápido progreso en los últimos años, dando impulso a las aplicaciones de sistemas de control realimentado descritos por Johansson y Jäntti (2010). Por ejemplo, las redes inalámbricas aumentan el ámbito del control de retroalimentación a los objetos móviles como los vehículos aéreos no tripulados (UAV) o las redes de inteligencia ambiental

(Litz *et al.*, 2005), y permiten la colocación de actuadores y sensores en lugares que son difíciles de acceder. En este contexto, las redes inalámbricas no solo reducen los costes de mantenimiento e instalación, sino que también aumentan la disponibilidad y la escalabilidad del sistema.

Así, el uso de redes de datos inalámbricas en los sistemas de control de realimentación crea desafíos significativos. En contraste con los buses de campo deterministas, estas redes tienen características de transmisión estocásticas. Esto significa que no solo se pueden producir pérdidas de transmisión, como se observa en las redes inalámbricas en particular, sino que también las transmisiones de datos pueden estar sujetas a retrasos desconocidos y variables en el tiempo. Por otra parte, si la red es compartida con otras aplicaciones, varias fuentes compiten por el acceso a la red, lo que implica restricciones adicionales para el sistema de control. La experiencia ha demostrado que los efectos de red inducidos pueden tener un impacto negativo en el rendimiento de un sistema de retroalimentación. Por ejemplo, incluso pequeñas variaciones de los tiempos de transmisión pueden desestabilizar un sistema de control (Bhuyan *et al.*, 2012).

Por lo tanto, surge la necesidad de nuevas estrategias de control y herramientas de análisis que sean capaces de hacer frente a los efectos de red no deterministas. La investigación en esta dirección se inició a principios de la década de 2000 y pronto se hizo conocido el concepto "sistemas de control en red" (NCS). Una característica de la investigación NCS es que tiene una complejidad compuesta, tanto para los ingenieros de control como para los ingenieros de telecomunicaciones, debido a que el objetivo común es la combinación de la teoría de control —que describe sistemas dinámicos conectados por enlaces perfectos— con el área de la teoría de la comunicación —que se ocupa de la información transportada a través de enlaces imperfectos—.

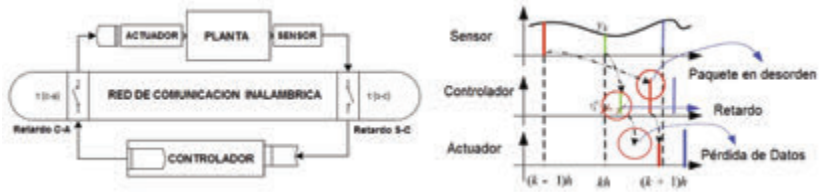
Control a través de la red de comunicación

Se sabe que la teoría de control clásica se basa en el hecho de que la comunicación entre sensores, actuadores y controladores es ideal,

lo que significa que los datos se comunican y se calculan con retrasos nulos o fijos, pero debido a las limitaciones físicas, cualquier medio de comunicación digital induce un retraso distinto de cero y no periódico. Para minimizar los efectos de los retrasos propios de las redes de comunicaciones se tienen dos opciones: la primera es optar por configurar una red dedicada; la segunda opción es el desarrollo de la teoría y el uso (WNCS) para ser capaz de especificar en qué condiciones más lentas y menos costosas, el hardware se puede utilizar de forma fiable en el sentido de dejar de garantizar un comportamiento adecuado de circuito cerrado. En este texto se sigue la segunda opción, ya que se presentan los resultados de la implementación de un WNCS, contribuyendo al desarrollo de esta teoría.

La teoría de control en red estudia las propiedades de los sistemas de control, como se puede observar en la figura 2, donde se ve que todas las aplicaciones de control realimentado en tiempo real y la comunicación entre sensores actuadores y controladores no es perfecta, mucho menos si esta comunicación se realiza mediante WNCS.

Figura 2
Estructura de una WNCS (izquierda)
y retardos en la comunicación de datos (derecha)



Fuente: los autores

Varios efectos de red inducidos se pueden introducir en la realimentación del lazo cerrado cuando se utiliza comunicación inalámbrica como:

- La presencia de un medio de comunicación compartido
- Modificar los intervalos de muestreo/transmisión
- Variación de los retardos de transmisión
- Abandonos de paquetes
- Cuantificación

Se conoce que cualquiera de estos fenómenos degrada el rendimiento e incluso pone en peligro la estabilidad del lazo cerrado (*cf.* Wei *et al.*, 2001; Hespanha *et al.*, 2007). Dependiendo de la red o la aplicación a realizar, la influencia e importancia de cada uno de los efectos mencionados puede variar de forma significativa. Así, en los sistemas que son capaces de transmitir los datos del sensor en un paquete no sufren de un medio de comunicación compartido, pero podría tomar más tiempo para recoger todos los datos y preparar la transmisión del paquete, lo que resulta en tiempos de retardo dominantes. No así en el caso en que los sensores comparten un bus CAN a través de un cable, en donde la información de los sensores es compartida pero los retardos de paquetes y peor aún las pérdidas se producen muy rara vez, lo que significa alta calidad de servicio de la infraestructura de comunicación en el bus CAN.

Un ejemplo es el sistema Beam-Ball o el control de voltaje de un motor-generador DC, en los cuales realizar la configuración de la comunicación, estabilizar el sistema, tomar la información del sensor, realizar los algoritmos respectivos en el controlador y nuevamente enviar esa información al actuador, es un proceso donde se presentan todos los efectos de los sistemas realimentados en red inalámbrica antes mencionados. Es por esta razón que se desarrolla e implementa un controlador PID predictivo basado en eventos capaz de estabilizar el sistema.

En este trabajo concentramos nuestra atención en tres efectos. El primer efecto —la presencia de un medio de comunicación común— requiere un protocolo de comunicación que relacione los datos con el control; aunque otros autores han considerado el caso cuando el protocolo es estocástico (*cf.* Antunes *et al.*, 2015), suponemos que el protocolo de comunicación es un proceso determinista, como también es asumido

por Donkers (*et al.*, 2011) y Walsh (*et al.*, 2002). Los dos siguientes efectos —la variación de los intervalos de transmisión y diferentes retardos de transmisión—, por lo general, no se consideran deterministas.

Algunos autores han centrado su estudio en el caso en donde la información estocástica está incluida en los modelos de los retrasos y los intervalos de transmisión a través de una función de distribución de probabilidad (Donkers *et al.*, 2012) o una cadena de Markov (Shi y Yu, 2009). En este proyecto centramos nuestra atención en el caso "distribución de probabilidad libre", como la sumatoria de retardos que también es estudiado por Cloosterman (*et al.*, 2010). En este caso, los intervalos de transmisión variables en el tiempo, y la incertidumbre y los retrasos de transmisión se toman de un conjunto acotado, sin pretender ningún conocimiento sobre la distribución de probabilidad particular. Por supuesto, si se puede obtener un modelo estocástico fiable y preciso de los canales de comunicación, siempre es más beneficioso usar esta información en el modelo y el análisis; sin embargo, en general, un modelo estocástico exacto puede ser difícil de obtener. En cualquier caso, tener una variedad de herramientas disponibles que tengan en cuenta los diferentes supuestos del modelo de red es beneficioso, ya que las redes exhiben comportamientos muy diferentes dependiendo de las configuraciones de red y los estándares de la red.

Control basado en eventos

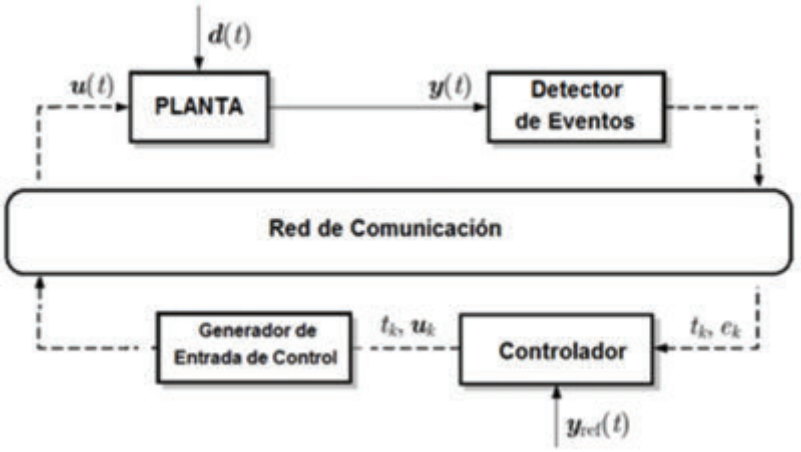
El control basado en eventos se define cuando el lazo de realimentación está cerrado solo si un evento indica que el error de la señal controlada excede de un límite tolerable y desencadena una transmisión de datos desde los sensores a los controladores y los actuadores. Por lo tanto, el control basado en eventos es un método importante para la reducción de la carga de comunicación de una red digital. A continuación se presentan las ideas principales de control basado en eventos.

Introducción al control basado en eventos

Control basado en eventos es una metodología de control que está siendo desarrollado como un medio para reducir la comunicación entre los sensores, el controlador y los actuadores en un lazo de control. Los instantes de muestreo no están determinados periódicamente por un reloj, sino por un detector de eventos, que se adapta el flujo de información en el lazo de retroalimentación con el comportamiento actual del sistema de lazo cerrado. Una comunicación entre los componentes se invoca solo después de que un evento ha indicado que el error supera un límite tolerable. Este principio de funcionamiento difiere respecto a la retroalimentación de datos muestreados en donde los datos de los sensores se comunican con el controlador en tiempos de muestreo periódicos, es decir, la comunicación se lleva a cabo independientemente de la magnitud del error; en estas situaciones, los recursos de comunicación e informáticos se utilizan innecesariamente.

En la figura 3 muestra los principales componentes de un lazo de control basado en eventos, en donde: la planta tiene una salida en el tiempo continuo $y(t)$, una entrada de tiempo $u(t)$ y el estado $x(t)$, la señal $u(t)$ es dada por el generador de entrada de control, $x(t)$ es continuamente evaluado por el detector de eventos. Los enlaces de comunicación representados por las líneas de trazos solo se utilizan después de que el detector de eventos ha indicado que el error e_k supera un límite tolerable dado en algún tiempo t_k, donde un evento dado por el estado actual $x(t_k)$ se transmite al controlador. El controlador determina la nueva entrada de u_k, que se utiliza en el generador de entrada de control para determinar la salida continua $u(t)$ en el intervalo de tiempo $[t_k, t_{k+1})$ hasta el próximo evento que se produce en el momento t_{k+1}.

Figura 3
Lazo de control basado en eventos



Fuente: los autores

En el control basado en eventos se rompe el paradigma del supuesto fundamental de la teoría de control para datos muestreados que genera un esquema de disparo periódico invocado por un reloj. Por lo tanto, para el control basado en eventos los modelos de tiempo discreto bien conocidos de la planta, el controlador y el sistema de circuito cerrado no se pueden aplicar, por ende, una nueva teoría debe ser desarrollada, que tenga en cuenta el comportamiento de los componentes asíncronos. Los nuevos objetivos principales de análisis y diseño de esta nueva teoría se refieren a la elección del generador de eventos y del generador de entrada de control. Así, el detector de eventos determina:

- Los instantes de tiempo t_k, (k = 0, 1,…) en el que se invoca la siguiente comunicación entre el detector de eventos, el controlador y el generador de entrada de control.
- La información que se comunica desde el sensor hacia el controlador.

El generador de entrada de control determina la señal $u(t)$ de forma continua durante el intervalo de tiempo $t \in [t_k, t_{k+1})$ en función de la información obtenida en t_k tiempo.

Las principales preguntas que hay que responder son:

- ¿En qué tiempo t_k en un lazo de realimentación se tiene que cerrar mediante el uso de los enlaces de comunicación?
- ¿Cuál es la información que el error e_k debería comunicar?
- ¿Cómo el generador de entrada de control debería determinar la entrada de control $u(t)$ entre dos eventos sucesivos?

Se han propuesto varios métodos para el control basado en eventos en los últimos años, que se distinguen con respecto a las respuestas dadas a estas preguntas. Algunos de ellos han sido publicados bajo diferentes nombres como el "control manejado por evento", "control activado por eventos", el "muestreo de Lebesgue", "control de banda muerta" o "envió de un delta de control". Las encuestas y las introducciones a estas técnicas se pueden encontrar en los trabajos de Wang y Lemmon (2011) y Tabuada (2007).

Existen varios motivos o escenarios de aplicación para el uso de retroalimentación basado en eventos:

- Cuando el intercambio de información en el lazo de realimentación se debe reducir a la comunicación mínima que es necesaria para asegurar un rendimiento de sistema requerido. Si la información se transfiere hacia y desde el controlador en una red de comunicación digital inalámbrica, un flujo de información reducida disminuye el riesgo de una sobrecarga de la red. Para nodos inalámbricos, la actividad reducida ahorra energía.
- Cuando existen sistemas en los que la estructura física requiere que las medidas o acciones de control sean tomadas en instantes de tiempo que dependen de la dinámica de la planta. Por ejemplo, cuando existe medición de elementos de rotación y en base a su posición se toman acciones de control.

- Cuando los protocolos de comunicación asíncrona y software en tiempo real hacen que no se permita transferir y procesar información en momentos específicos del reloj, sino conducir a un comportamiento asíncrono de todos los componentes de un circuito de retroalimentación. Los sensores y actuadores del mismo modo se activan por los acontecimientos, ya que trabajan si llega alguna información nueva. Para este tipo de componentes el esquema de trabajo basado en eventos es más "natural" que el muestreo periódico.

Propiedades fundamentales de control basado en eventos

El principio de funcionamiento orientado a eventos implica que la entrada a la planta $u(t)$ se determina por una combinación de control predictivo y de control de realimentación. En los tiempos de eventos t_k, la entrada $u(t_k)$ depende de un modo de lazo cerrado del estado actual $x(t_k)$ (siempre que los enlaces de comunicación no introduzcan un retardo de tiempo considerable), mientras que entre los dos tiempos de eventos t_k y t_{k+1} la entrada $u(t)$ se genera como control en lazo abierto en función de datos pasados de u_k.

El objetivo del control basado en eventos es muestrear solo si se debe evitar una degradación severa del rendimiento. La mayoría de los esquemas de control basados en eventos no pueden garantizar la estabilidad asintótica del sistema de circuito cerrado. En cambio, el estado de la planta $x(t)$ debe mantenerse en los alrededores de Ω del estado de equilibrio $x^{\wedge}$. "La propiedad $x(t) \in \Omega$ para todo $t \geq T$ se llama la acotación final o estabilidad práctica del sistema en lazo cerrado. Por lo general, el tamaño del conjunto Ω depende del límite de umbral del e_k" (Lunze, 2014).

Comparación entre control basado en eventos y control de datos muestreados

Existe una diferencia entre el control basado en eventos y el control de datos muestreados, esto lo demuestran los resultados analíticos obtenidos para los sistemas de primer orden presentados en varios tra-

bajos (Åström, 2008; Åström y Bernhardsson, 2001; Lehmann *et al.*, 2012; Rabi *et al.*, 2008). Los trabajos citados evidencian que el muestreo basado en eventos, de hecho, puede conducir a una reducción considerable de la comunicación dentro del lazo de control. Además, si el sistema está muy perturbado, el lazo de control basado en eventos puede tener un mejor rendimiento que el lazo de datos muestreados, ya que en esta situación se invoca la comunicación con más frecuencia que el reloj.

Existen métodos que son aplicables para los sistemas de orden superior y diferentes métodos de generación de eventos de entrada-activación y control. Los métodos diferentes son:

- Realimentación de estado basado en eventos. El generador de entrada de control está construido de manera que el sistema de lazo cerrado imita el comportamiento de un lazo de realimentación de estado continuo con precisión ajustable. Se muestra que el generador de entrada de control tiene que incluir un modelo del lazo de realimentación de estado continuo.
- Control basado en eventos distribuido. Para los sistemas interconectados, la idea de realimentación de estado basado en eventos puede, o bien ser implementado como un controlador distribuido, lo que conduce al mismo rendimiento general del sistema como la retroalimentación centralizada, o bien pueden aplicarse por separado a los subsistemas aislados resultantes en un esquema de control basado en eventos descentralizados.
- Control basado en optimización. El controlador basado en eventos se puede obtener como la solución de un problema de control óptimo, si se divide el espacio de estados, y la mejor entrada constante posible se aplica siempre que el estado permanezca en la misma partición de espacio de estado.
- Control basado en eventos de sistemas estocásticos. Si se formula como un problema de optimización estocástica, el detector de eventos y el generador de entrada de control tienen que ser diseñados de manera simultánea, lo cual conduce a un problema de optimización muy complejo; sin embargo, ambos componentes

se encuentran en una estructura de información anidada, en la que el detector de eventos conoce las $x(t)$ del estado de la planta para todo el tiempo t y el generador de entrada de control solo conoce el estado $x(t_k)$ en un t_k tiempo del evento. El problema en general se puede descomponer en dos problemas de optimización anidados que se pueden resolver de forma secuencial.

De los métodos anteriormente descritos en este trabajo se realizó el control basado en eventos de realimentación de estado, pues el detector de eventos está dado en base al umbral del error de estimación y la planta solo transmite datos si el error sobrepasa el límite umbral; en cambio, el generador de entrada de control está en el lado del controlador y está implementado de manera que el sistema de lazo cerrado imita el comportamiento de un lazo de realimentación de estado continuo con precisión ajustable. El generador de entrada de control tiene que ejecutar un algoritmo predictivo en base a los datos recibidos por el detector de eventos el cual está en el lado del sensor, en donde se incluye el modelo estimado del controlador del lazo de realimentación de la planta de estado continuo.

Características de los WNCS

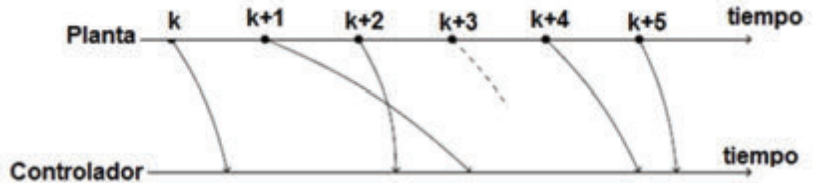
Los primeros sistemas de control en red son sistemas cableados, donde los sensores y controladores están conectados a través de un medio común cerrando cada uno de los muchos lazos de control con cables dedicados. Como uno de los parámetros más críticos para la estabilidad de un sistema de control es el retraso —como lo demuestran Kao y Lincoln (2004)—, el protocolo de red utilizado en tales sistemas de control debe satisfacer las restricciones de tiempo real. Algunos protocolos de cable que pueden trabajar bajo limitaciones de tiempo real son: Control de Red de Área (CAN) desarrollado para ser utilizado en la automatización de los vehículos y PROFIBUS diseñado para la fabricación y automatización de procesos. Estos sistemas con comunicación alámbrica están siendo reemplazados ampliamente en la industria por los WNCS, como lo afirma Bregulla (*et al.*, 2011), pero a pesar de que reducen el

cableado y todos los costos asociados con ello, existen problemas con los retardos y las pérdidas de paquetes aleatorias, que todavía no se resuelven satisfactoriamente. Por lo tanto, una gran cantidad de investigación se ha centrado en los sistemas de control inalámbrico, donde el cableado está ausente entre los elementos que componen el sistema a controlar como el sensor, el controlador y el actuador.

Sistemas inalámbricos: las imperfecciones de la red inalámbrica

Una de las desventajas más evidentes de utilizar el medio inalámbrico para la comunicación son los retardos de envió-recepción de los paquetes de información conocido como *jitter*. Para un sistema de control inalámbrico de tiempo, en el orden de los milisegundos, impulsado a intervalos regulares, la planta toma muestras y las muestras están programadas para ser entregadas al controlador. Sin embargo, debido a la naturaleza estocástica del medio se introducen retardos y el controlador recibe paquetes a intervalos irregulares, como se ilustra en la figura 4.

Figura 4
Transmisión inalámbrica irregular



Fuente: los autores

Los factores que contribuyen a este comportamiento son:

- Retardo de acceso de red: el tiempo necesario para que la red esté disponible para aceptar los datos.
- Tiempo de retardo del paquete de datos: el tiempo para que el transmisor coloque un paquete de datos en la red.

- Retardo de transmisión: el tiempo que el paquete pasa dentro de la red; la cantidad de estos retrasos depende de la anchura de banda del canal, tamaños de paquetes y pérdidas en general.

También hay otros problemas que se dan aparte del retardo de llegada de los paquetes al controlador, por ejemplo, hay un riesgo de paquetes que se pierden dentro de la red debido al ruido, la interferencia, fallos en los nodos y las colisiones. Cuando se detecta la pérdida de paquetes es muy probable que en el momento en que se vuelve a transmitir los datos sean obsoletos, por lo que es más útil transmitir un paquete nuevo en lugar del paquete antiguo. Esto prácticamente implica que el sistema debe tolerar la pérdida de paquetes.

Frente a la pérdida de paquetes, una aparente solución debería ser mantener el tiempo de muestreo bajo y el envío de forma redundante de muchos paquetes, de modo que el controlador pueda utilizar los que llegan a tiempo. Pero, dado el limitado ancho de banda del medio, el envío de paquetes a alta frecuencia dará lugar a más tráfico, por lo tanto más colisiones y retardos más largos. Además, los sensores inalámbricos generalmente funcionan con energía limitada para mantener su esperanza de vida larga, lo cual entraría en conflicto con el envío de forma redundante de muchos paquetes. Por lo tanto, se han preferido las soluciones con mayor tiempo de muestreo.

Acceso a la red

El acceso al medio debe ser regulado cuando existen transmisiones de paquetes de control, sobre todo para mantener el retardo de acceso a la red y las colisiones bajo control. Para ello se puede utilizar una forma determinista, la cual será planificar los tiempos cuando el nodo transmite, conocido como acceso múltiple por división de tiempo (TDMA), de manera que cada nodo tiene un periodo específico cuando se puede transmitir. Otra forma determinista es dividir la banda de frecuencia de manera que cada nodo tenga una frecuencia exclusiva para transmitir, esta técnica es conocida como división de frecuencia de ac-

ceso múltiple (FMDA). Estas formas de transmitir determinan mayor límite de retraso, pero por lo general son difíciles de aplicar debido a la construcción de un planificador y a la sincronización de los nodos de modo que puedan cumplir con la programación del planificador.

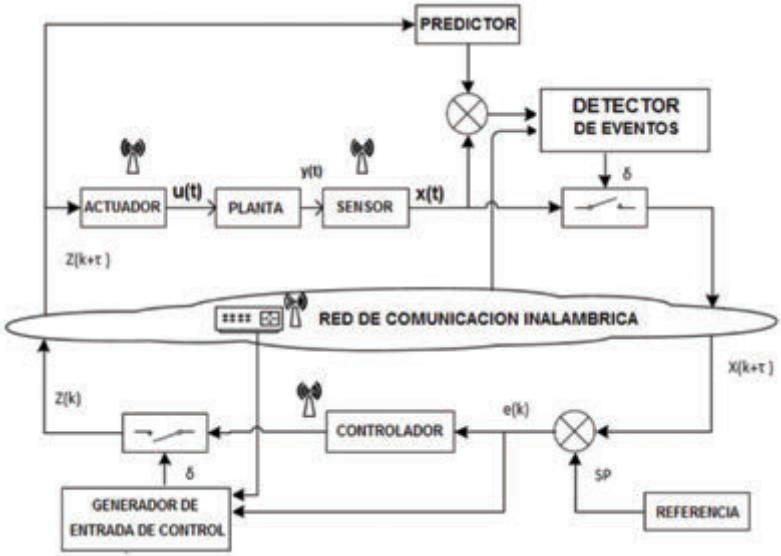
Otra forma aleatoria para acceder al medio es poner en práctica protocolos de acceso aleatorio, donde un nodo que tiene un paquete para transmitir primero comprueba si el medio está disponible y luego lo envía de inmediato si lo está. Si el medio está ocupado, el nodo espera durante una cantidad de tiempo aleatorio y vuelve a intentar enviar. Estos protocolos de acceso aleatorio, inevitablemente, aumentan el retardo de acceso a la red, lo cual los hace más difícil de ser empleados en los sistemas de control.

Al igual que en todas las redes de medio compartido, el control de acceso al medio (MAC) es una técnica importante que permite el funcionamiento exitoso de la red. Una tarea fundamental del protocolo MAC es evitar colisiones de nodos interferentes. Hay muchos protocolos MAC que se han desarrollado para las redes inalámbricas de voz y de datos. Entre estos se incluyen el acceso múltiple por división de tiempo (TDMA), el acceso múltiple por división de código (CDMA) y los protocolos basados en contención como IEEE 802.11.

Estructura del sistema

La representación de red compartida en la que los sensores, el controlador y el actuador están espacialmente separados unos de otros (figura 5) y la comunicación se realizan a través de la red. En esta arquitectura, los costos de la utilización de una red como retardo se introducen dos veces, una vez desde los sensores al controlador y otra desde el controlador al actuador. Habiendo considerado que los actuadores, generalmente, exigen alta potencia, esta arquitectura también es beneficiosa en términos de fuente de alimentación del controlador. Vale anotar que esta es la arquitectura que hemos utilizado en este trabajo.

Figura 5
Estructura directa de la WNCS



Fuente: los autores

Modelo de retardo

Existen diferentes propuestas para modelar el retardo inducido por la red y las pérdidas de paquetes. Una descripción la tenemos en el trabajo de Nilsson (*et al.*, 1998), de donde se ha extraído los siguientes modelos de retardo:

1. **Retardo Constante.** La forma más determinista para modelar el retardo de red es la determinación de un valor y utilizar este como el retardo constante predeterminado, que solo puede ser aplicable a casos deterministas. En este caso el valor medio del peor retardo puede ser utilizado como el retraso constante, para el análisis. Sin embargo, este tipo de seguridad disminuirá el rendimiento ya que el retraso de control puede ser innecesariamente

largo. Este modelo de retardo no está adaptado a las redes con protocolos de acceso al medio estocásticos:

$$\tau_1(t) = \tau_c$$

2. **Retardo con Distribución Randómica Gauseana.** Para realizar análisis y representar las variaciones en los protocolos de acceso aleatorio, se asume un modelo de distribución normal que representa el retardo, donde la media y la varianza se seleccionan para efectos de simulación del comportamiento de los retardos en la red:

$$\tau_2(t) = x_1(t), X_1 \sim N(\mu, \sigma)$$

3. **Retardo Aleatorio Parcialmente Independiente.** En una red los retardos son estocásticos y pueden tener varias fuentes, así por ejemplo:

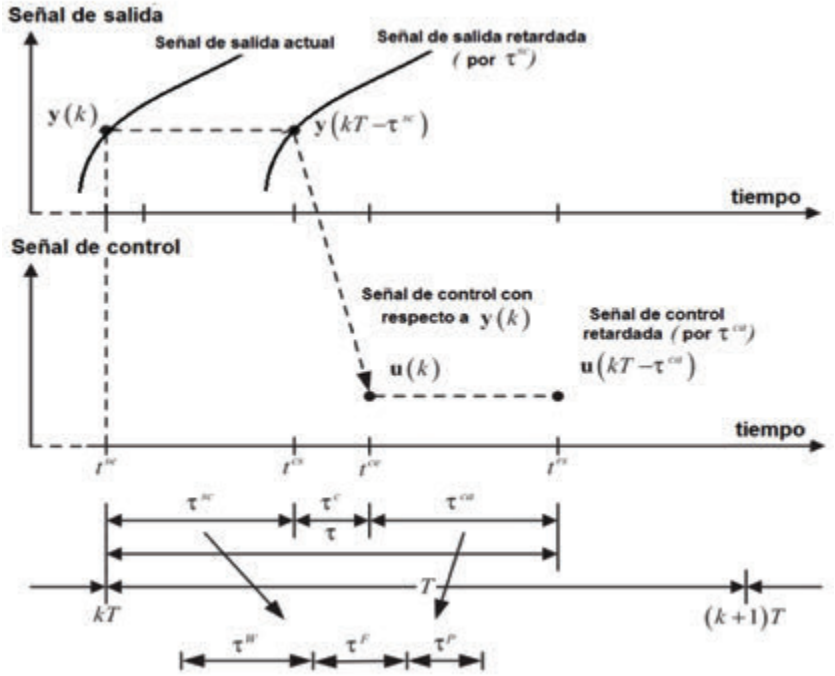
   - Tiempo de espera a que la red esté activa.
   - Tiempo de espera para la transmisión de los mensajes.
   - Si se producen errores de transmisión, puede ser necesaria una retransmisión.
   - Debido a colisiones, si dos nodos intentan enviar al mismo tiempo, la resolución de esto puede incluir una espera aleatoria.

Debido a que las actividades del sistema generalmente no están sincronizadas entre sí, el número de las causas de retardo enumeradas anteriormente que se producirán es aleatorio. Para tener en cuenta la aleatoriedad de los retrasos de red en el modelo, los retornos de tiempo se pueden modelar tomando como base una distribución probabilística. Para mantener el modelo simple de analizar se puede suponer que el retardo de transferencia sea independiente de los tiempos de retardo anteriores. En un sistema de comunicación real, sin embargo, el tiempo de transferencia se correlacionará usualmente con el último retardo de transferencia. Por ejemplo, la carga de la red, que es uno de los factores que afectan al retardo, varía típicamente con una constante de tiempo

más lenta que el periodo de muestreo en un sistema de control, es decir, el tiempo entre dos transferencias.

En este trabajo vamos a permitir que el modelo tenga diferentes distribuciones de probabilidad para el retardo desde el sensor al controlador, Tk^{sc}, y para el retardo desde el controlador al actuador, Tk^{ca}, como se indica en la siguiente figura:

Figura 6
Diagrama de retardos causados en la red

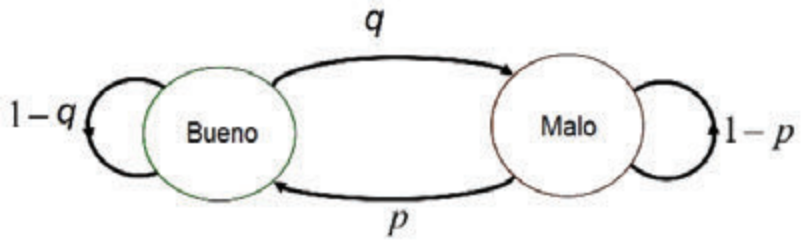


Fuente: los autores

Modelo de abandono de paquetes

En el caso de abandono o pérdida de paquetes, la forma más simple de modelar es tratarlo como un proceso de Bernoulli, donde la variable "pérdida de paquetes" es única e independiente. Cuanto mayor es la probabilidad de transmisión mejor es la condición de la red, sin embargo, se ha observado que las pérdidas de paquetes generalmente aparecen en ráfagas (Willig *et al.*, 2002), por tanto, no es una buena aproximación ignorar esta correlación y el paquete de modelo temporal de deserción por un modelo de pérdida independiente. En lugar de esto, un método más completo es el modelo de canal Gilbert-Elliot, que es un modelo de cadena de Markov de dos estados, compuesto de estados "buenos" y "malos". En el estado "bueno" los paquetes se entregan sin un error y la tasa de fracaso q es la probabilidad de la transición al estado "malo", en el que se pierden los paquetes. La probabilidad de la transición al estado "bueno" se denota por p. Este modelo, representado en la figura 7, captura la correlación temporal de las pérdidas de paquetes y se puede extender a las cadenas de Markov con más estados si es necesario.

Figura 7
Modelo de pérdida de paquetes Gilbert-Elliot



Fuente: los autores

Capítulo dos
Controlador basado en eventos

Sistemas basados en eventos

Los sistemas de control que se han utilizado hasta ahora en muchas áreas de la ingeniería se han basado en un muestreo periódico. Para dichos sistemas existe toda una teoría probada y muchas herramientas matemáticas dando resultados teóricos bien establecidos; en cambio, el muestreo basado en eventos aparece de manera natural en las redes de sensores inalámbricos (WSN), donde los sensores envían sus salidas en función de alguna condición. Además, las estrategias basadas en eventos se han utilizado en áreas tales como el control de los procesos industriales (Kwon *et al.*, 1999), la planificación de la trayectoria del robot (Tarn *et al.*, 1996) y el control del motor (Hendricks *et al.*, 1994). Sin embargo, el control basado en eventos ha sido utilizado principalmente de forma *ad-hoc*, debido a la falta de resultados teóricos, los cuales recién han comenzado a estar disponibles, por ejemplo, en los trabajos de Pawlowski (*et al.*, 2008), Åström y Bernhardsson (2011) y Hespanha (*et al.*, 2007). Asimismo, últimamente el control basado en eventos se está utilizando en sistemas distribuidos (Weimer *et al.*, 2012).

Los sistemas basados en eventos frecuentemente se clasifican en dos tipos, los sistemas activados por eventos (Yu y Antsaklis, 2011) y los sistemas autodisparados (Almeida *et al.*, 2015; Tabuada, 2007). En los primeros, por lo general no se tiene un modelo de la planta, donde los cambios medidos en el estado provocan cambios esporádicos del controlador. En los sistemas autodisparados el controlador calcula el momento en que se activa el siguiente evento en base a una estimación del

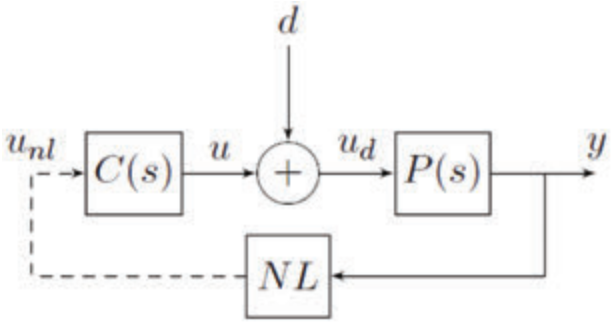
estado de la planta obtenido a partir de una función de estimación que tiene que ver con el comportamiento del proceso. Para el desarrollo de este trabajo se utilizó los sistemas autodisparados.

Algunos trabajos relacionados con los sistemas activados por eventos (Postoyan *et al.*, 2013; Heemels y Donkers, 2013) utilizan una estrategia periódica activada por eventos, en la que la condición de evento es verificada en un periodo de muestreo constante. En Lehmann (*et al.*, 2012) se analiza la relación entre el control de datos muestreados y dos estrategias activadas por eventos, a saber, el control de banda muerta y el control activado por eventos basado en modelos, concluyendo que el enfoque basado en eventos garantiza la mejor estabilidad y aprovecha las propiedades de comunicación. Por esta razón, la industria ha volcado sus esfuerzos en aprovechar las redes inalámbricas de comunicación para el control de muchos procesos, estandarizando el uso de controladores, buscando realizar implementaciones de controladores PID basados en eventos, como se ve en los trabajos de Årzén (1999) y Vasyutynskyy y Kabitzsch (2006). Además, un estudio exhaustivo de los diferentes métodos propuestos en la literatura para el control PID basado evento puede encontrarse en el texto de Sánchez (*et al.*, 2012).

Este proyecto se centra en la implementación de un controlador PID basado en el muestreo activado por eventos y en particular sobre el paso a nivel o el muestreo *send-on-delta*, que consiste en tomar una nueva muestra cuando un cambio mayor que un umbral definido δse detecta en la señal. En la práctica, este sistema es equivalente a introducir una no linealidad (NL) que se puede caracterizar como una cuantificación de N niveles con histéresis. El comportamiento de un esquema de control basado en el muestreo del paso a nivel se estudió considerando que la toma de muestras se encuentra después de la salida del proceso, el cual presenta una configuración de un esquema de control basado en las transmisiones inalámbricas (figura 8) y tiene diferentes propiedades: en los casos estudiados corresponde a una planta ($P(s)$ = Beam-Ball) o al control de voltaje de un motor-generador DC con un sensor inalám-

brico que toma medidas de la variable de proceso (y = distancia o y = voltaje de salida) y los envía al controlador ($C(s)$), separado físicamente del sensor. Luego los datos del sensor son procesados en el controlador y los resultados de este (u) son enviados al actuador, que también está físicamente separado, pero conectado a la planta.

Figura 8
Esquema de un sistema de control basado en eventos



Fuente: los autores

En este capítulo se caracteriza con un enfoque sistemático el comportamiento del control basado en eventos con el proceso del controlador PID, considerando el retardo de tiempo; además, se pretende dar una alternativa para el análisis y aplicación de controladores PID basado en eventos mediante comunicación inalámbrica. La obtención de los resultados ha sido confirmada con la práctica.

Muestreo basado en eventos

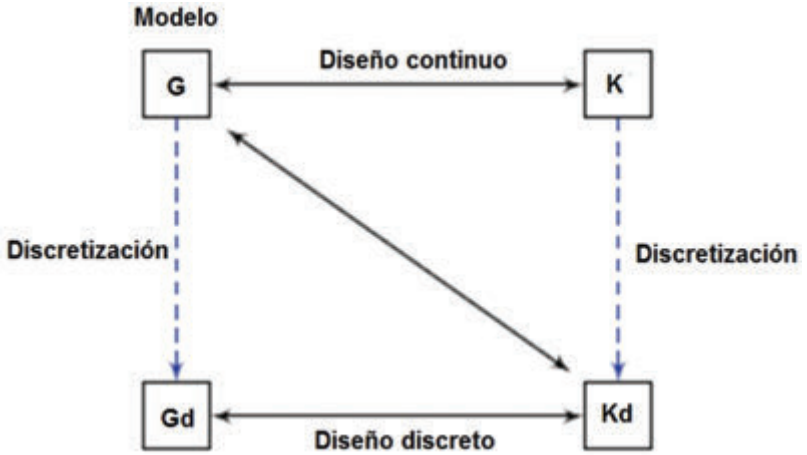
Sistemas de datos muestreados

Es conocido que una gran mayoría de sistemas en ingeniería son continuos en la naturaleza, pero gracias a la capacidad de las computadoras digitales para procesar datos discretos, los sistemas de tiempo continuo se controlan usando observaciones muestreadas tomadas en

instantes discretos. Por lo tanto, los sistemas de control resultantes son un híbrido, que consiste en interactuar con componentes discretos y continuos como se muestra en la figura 9. Estos sistemas híbridos, en los que el sistema a controlar evoluciona en tiempo continuo y el controlador evoluciona en tiempo discreto, se denominan sistemas de datos muestreados, como lo describe Xu (2015).

La característica significativa del diseño de sistemas de datos muestreados que lo distingue de las técnicas estándar para el diseño de sistemas de control, es que debe enfrentarse con modelos de plantas y leyes de control que se encuentran en diferentes dominios. Existen tres metodologías principales para el diseño y análisis de sistemas de datos muestreados que están representados gráficamente en la figura 9, donde G es un proceso de tiempo continuo y Kd es una ley de control de tiempo discreto. Los tres métodos comienzan con el principio del modelo de tiempo continuo G y apuntan a diseñar el controlador de tiempo discreto Kd y analizar su rendimiento.

Figura 9
Enfoques de sistemas de datos muestreados



Fuente: los autores

Las dos aproximaciones bien conocidas siguen los caminos alrededor del perímetro del diagrama. La primera consiste en realizar todo el análisis y diseño en el dominio de tiempo continuo utilizando un sistema que se cree que es una aproximación cercana al sistema de datos muestreados. Esto se logra asociando cada controlador de tiempo continuo K con una aproximación de tiempo discreto Kd mediante un método de discretización. La síntesis y análisis del controlador se realizan, entonces, en tiempo continuo, con la suposición subyacente de que el comportamiento del sistema de lazo cerrado obtenido con el controlador K refleja de cerca lo conseguido con la implementación de datos muestreados Kd. Por lo tanto, este método no aborda directamente la cuestión de la implementación en la etapa de diseño. La segunda comienza discretizando el sistema de tiempo continuo G, dando una aproximación de tiempo discreto Gd, por tanto, ignorando el comportamiento entre muestras. Entonces, el controlador Kd está diseñado directamente en tiempo discreto usando Gd, con la creencia de que el rendimiento de este sistema de tiempo puramente discreto se aproxima al del sistema de datos muestreados.

Finalmente, un tercer enfoque ha atraído considerable actividad de investigación por su facilidad en la implementación. En este enfoque, el sistema G y la interconexión Kd del controlador se tratan directa y exactamente. Nuestro trabajo en este texto se centra, en la medida de lo posible, en este enfoque, mientras que en algunos casos utilizaremos el segundo enfoque para explicar más sencillamente el diseño del controlador.

Las técnicas de muestreo y control basadas en eventos

En los últimos años los investigadores que integran tecnologías en varias áreas han desarrollado estas técnicas. La reducción de la información en el intercambio entre los agentes que intervienen en un lazo de control —sean sensores, controladores y actuadores— es de suma importancia, más aún cuando hay limitaciones en el flujo de informa-

ción debido a que los datos se intercambian por las redes inalámbricas, por lo que reducir la carga de tráfico es clave, pues cuanto más tráfico hay mayor es la posibilidad de pérdida de datos y se puede experimentar retardos aleatorios; así, una gran reducción en el tráfico es una problema esencial en las redes inalámbricas, si mayor es la reducción en el flujo de información, mayor será la disminución de las operaciones y transmisiones de cálculo y por tanto más largo es el tiempo de vida de las baterías que abastecen de energía al sistema.

Es importante definir previamente un "evento": es algo que ocurre en un determinado tiempo, que se produce o se dispara cuando ocurre alguna condición booleana o lógica, por ejemplo una condición genérica basada en eventos se representa como (*Condición Lógica Verdadera*) o $(t_{\text{evento}} \geq t_{\text{max}})$ $(Condición\ lógica\ es\ verdadera)$ó $(t_{evento} \geq t_{max})$.

La condición lógica se detecta si algo pasa en un momento determinado, y por lo tanto se considera el término asíncrono de la expresión. Para el control de procesos, la condición lógica se compone de operaciones booleanas, donde las variables son las señales que el sensor recibe del proceso o la acción de control producida por un controlador. Esto es utilizado debido a que hay situaciones en las que la condición lógica no puede ser verdad y, por lo tanto, no se pueden producir eventos.

Dado que la activación de un evento en el control de procesos significa, en general, la activación de un controlador para realizar una o varias acciones de control en función de la estrategia de control, un excesivo número de eventos podría generar un gran número de acciones de control que posiblemente puedan dañar los actuadores o en su defecto saturar los canales de comunicación en el lazo de control, produciendo retrasos y abandonos de datos. Para poder evaluar y detectar el momento exacto en el tiempo cuando una condición lógica se cumple, se debe evaluar de forma continua en tiempo. Por lo que, una evaluación continua de la condición implica un muestreo continuo de las variables o funciones implicadas en la expresión lógica; pero los sistemas de control digitales actuales implementan estrategias de

muestreo discretos síncronos. Por esta razón, los enfoques prácticos de control basado en eventos se implementan para muestrear las variables y evaluar las condiciones basadas en eventos tan rápido como sea posible. Esto se conoce como el "enfoque compuesto", donde los eventos asíncronos se pre sincronizan por la vía rápida tomando muestras periódicas (Miskowicz, 2006).

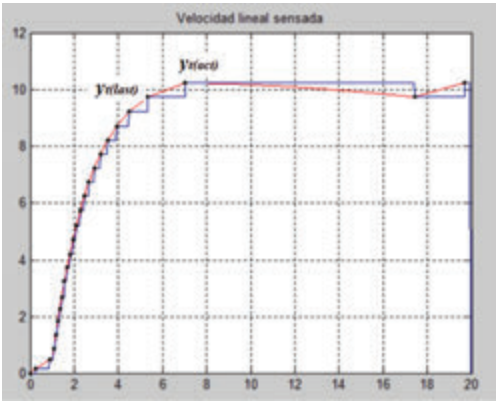
En la figura 10 se presenta el resultado de la evaluación de una simple condición basada evento usando una señal continua obtenida, por ejemplo, de un sensor incorporado en un proceso industrial. En este ejemplo, la condición lógica asíncrona para evaluar es:

$$|y(t_{act})\text{-}y(t_{last})| \geq \Delta \tag{2.1}$$

Donde $y(t_{act})$ es el valor actual de la salida del proceso $y(t)$, $y(t_{last})$ es el valor previo de la condición verdadera de $y(t)$, y Δ se conoce como el umbral o delta, que asume un valor menor que el rango dinámico de $y(t)$. Esto se expresa como $\Delta < |y(max) - y(min)|$ $\Delta < |y(max)\text{-}y(min)|$. Las señales resultantes de la figura 10 se obtienen mediante la simulación de las señales de salida del proceso actual y anterior, cuando se actualiza la condición lógica generando una retención de orden cero (ZOH).

El proceso de generación de una señal discreta en el tiempo mediante la evaluación de una condición basada en eventos sobre una señal continua en el tiempo, se conoce comúnmente como "muestreo basado en eventos". Otros nombres que se encuentran en la literatura de control son: muestreo de frecuencia variable (Dorf, 1962), muestreo adaptativo (Laurent *et al.*, 1969), muestreo de banda muerta (Otanez *et al.*, 2002), muestreo de Lebesgue (Åström y Bernhardsson, 2011), muestreo *sent-on-delta* (Miskowicz, 2006) y muestreo por cruce de nivel (Miskowicz, 2005).

Figura 10
Muestreo basado en eventos



Fuente: los autores

Un resumen de las estrategias de muestreo basado en eventos se muestra en la tabla siguiente:

Tabla 1
Condición lógica utilizada para el muestreo por eventos

Condición Lógica	Verdadero cuando
$\lvert y(t)\text{-}y(t_{last})\rvert \geq \Delta$	La diferencia entre la salida actual y la salida anterior es mayor que Δ
$\int_{tant}^{tact} \lvert y(t)\text{-}y(t_{last})\rvert \geq \Delta$	La integral de la diferencia entre la salida actual y la salida anterior entre el tiempo actual y anterior es verdadero cuando es mayor que Δ
$\lvert \hat{y}(t)\text{-}y(t)\rvert \geq \Delta$	La diferencia entre el valor estimado de la señal de salida y la salida es mayor que Δ
$\int_{tant}^{tact} \lvert \hat{y}(t)\text{-}y(t)\rvert \geq \Delta$	La integral de la diferencia entre el valor estimado de la señal de salida y la señal de salida es verdadero cuando es mayor que Δ
$\int_{tant}^{tact} \lvert y(t)\text{-}y(t_{last})\rvert^2 \geq \Delta$	La integral del cuadrado del error entre la salida actual y la salida anterior entre el tiempo actual y anterior es verdadero cuando es mayor que Δ

Fuente: los autores

¿Cómo funciona el muestreo basado en eventos en el marco del control automático de procesos? En los sistemas de control digital, las acciones de los agentes de control (sensores, controladores, actuadores) se rigen por una señal de reloj común. Esto da lugar a las acciones sincronizadas de más de un intervalo (h_{nom}) de muestreo que incluyen la toma de muestras del proceso por el sensor, la activación del controlador y la aplicación de la acción de control al actuador. Esta es la base para el muestreo de los sistemas de control. Pero si se aplica el paradigma basado en eventos, diferentes eventos independientes causarían los agentes de control para llevar a cabo las acciones de una manera asíncrona. Una condición basada en eventos podría ser diferente para cada agente, para detectar un cambio en una variable de sistema o una función de otra variable del sistema. El ejemplo de esta situación se da cuando un sensor detecta un cambio en una señal (el evento) y envía una muestra del proceso de salida al controlador (la acción).

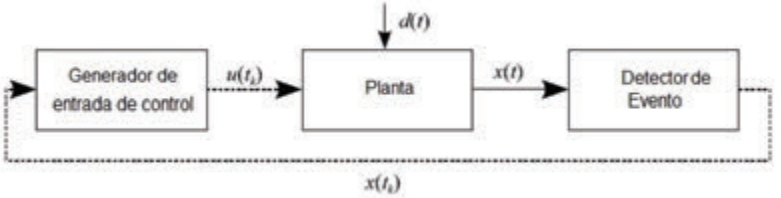
A continuación se realiza una clasificación general de las estrategias de control basadas en eventos que se ha encontrado en la literatura; además, se revisa la evolución de control PID basado en eventos. Cabe señalar que, según nuestros conocimientos, no existen implementaciones de controladores PID industriales basados en eventos y hay pocas publicaciones con resultados apoyados por procesos reales (*cf.* Lehmann y Lunze, 2011; Guarnes *et al.*, 2009; Sánchez *et al.*, 2011).

Estrategias de control basado en eventos

Últimamente se han dado investigaciones en relación con el muestreo y control basado en eventos, especialmente las relacionadas con los sistemas de control en red inalámbrica. En nuestro trabajo, los agentes pueden poseer los eventos de entrada para recibir información entre ellos, por ejemplo: el controlador recibe información de la entrada del sensor, el actuador recibe información desde el controlador, los eventos para activar la transmisión de información dependen de la condición lógica utilizada para el muestreo por eventos, así, la información

de una muestra del sensor va al controlador o la información procesada por el algoritmo de control va al actuador, permitiendo realizar la acción de control. Los controladores pueden tener diferentes algoritmos de control. Para proporcionar un esquema básico de los PID basado en eventos, partimos de una estructura simplificada del sistema con control basado en eventos presentado por Åstrom (2008), que se muestra en la figura 11. Allí se considera un sistema genérico: una planta, un detector de evento y un generador de entrada de control.

Figura 11
Diagrama de bloques simplificado de un WNCS



Fuente: los autores

El detector de evento envía los datos de salida del proceso, vector de estado para activar el generador de entrada de control cuando se produce un evento. El generador de entrada de control está inactivo entre dos eventos consecutivos y se activa mediante una invocación del detector de eventos (también es posible activarlo mediante una señal de tiempo, pero esto implicaría un estimador del estado de la planta hasta cuando exista nueva información disponible). Por lo tanto, el generador de entrada de control funciona en un lazo cerrado solo para los eventos (retroalimentación). Del mismo modo, una vez que en el generador de entrada de control se ha producido la nueva acción de control, hay dos condiciones posibles que pueden ser utilizados para enviar los datos al actuador: cuando están disponibles o cuando una condición lógica situada en su salida es verdadera, es decir, una solución basada en eventos. Dependiendo de la condición basada en eventos ubicados en el detector de evento y el algoritmo de control incluido en el generador de entrada de control, se pueden distinguir tres categorías genéricas:

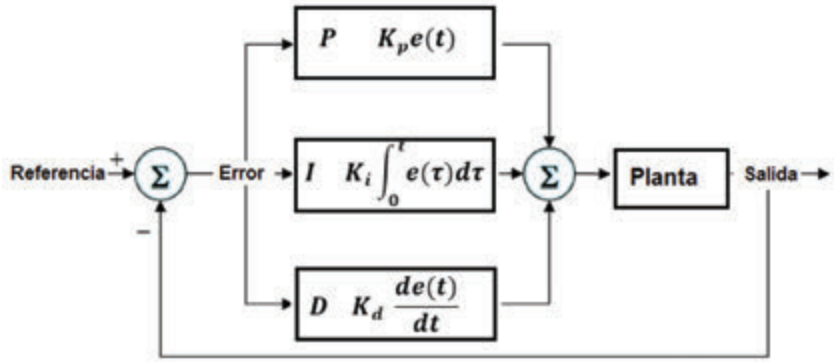
- En la primera, las acciones de control de realimentación se calculan cuando alguna variable varía más que un cierto valor (para diferentes condiciones basado en eventos ver la tabla 2). Por lo tanto, el generador de entrada de control produce una nueva acción de control cada vez que $x(t_k)$ cruza hacia arriba o abajo los niveles calculados de antemano (, -3Δ, -2Δ, -Δ, 0, Δ, 2Δ, 3Δ,…). La ley de control empleado en esta categoría es, en la mayoría de los casos, un PID en periodo variable de muestreo (Pawlowski *et al.*, 2008; Årzén, 1999; Durand *et al.*, 2010), aunque hay enfoques basados en la retroalimentación de estado (Heemels *et al.*, 2008; Henningsson y Cervin, 2009; Rabi y Baras, 2007).
- En el segundo grupo, la acción de control se ajusta al máximo/mínimo, mientras que $x(t)$ se queda fuera de una determinada banda muerta alrededor del *set point ysp* o una trayectoria de estado X, es decir, mientras que $|y(t_{act}) - ysp| \geq \Delta$ o $X(t_{act}) - Xs \geq \Delta$; en el mismo instante $x(t)$ vuelve a entrar en la banda, surge un nuevo evento, y el generador de entrada de control produce una señal de control de alimentación directa para mover el proceso asintóticamente al valor de referencia o de la trayectoria (Cervin y Åström, 2007; Lehmann y Lunze, 2011; Åstrom, 2008). La señal de control de alimentación directa se obtiene por medio de una retención generalizada que se pone a cero cada vez que x vuelve a entrar en la banda.
- Por último, es posible definir un tercer grupo de enfoques basados en eventos, donde se genera una acción de impulso para llevar el proceso al valor de consigna o al origen cuando X cruza los límites de la banda muerta; el resto del tiempo se pone a cero la acción de control (Åström y Bernhardsson, 2011; Cervin y Henningsson, 2008; Rabi y Johansson, 2009).

Para resumir, el primer grupo mantiene una ley de control constante durante el periodo de muestreo no uniforme, delimitada por dos eventos consecutivos, el segundo grupo es una ley de control variable y el tercer grupo es uno impulsivo. Nuestro proyecto se enmarca en el primer grupo.

El controlador PID basado en eventos

Tomando en cuenta los diferentes tipos de políticas de control basadas en eventos que se han presentado anteriormente, ahora se aborda el tema central de este capítulo. A continuación, se da una descripción de los controladores basados en eventos bajo la ley de control de PID (figura 12) implementados en el generador de entrada de control.

Figura 12
Diagrama de bloques y representación simplificada de un controlador PID



Fuente: los autores

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt + K_p T_d \frac{de(t)}{dt} \qquad (2.2)$$

Para la implementación digital de este controlador, se debe discretizar la ley de control. Esto se consigue normalmente utilizando diferencias hacia adelante para el término integral de manera que es posible pre calcular la parte integral de tiempo t_{k+1} y el uso de las diferencias hacia atrás para la parte de derivación. Por lo tanto, al indicar el intervalo de muestreo como h_{nom}, el siguiente pseudo código de este controlador PID es como se

indica a continuación: el error es la diferencia entre el punto de referencia menos la señal de salida del sistema dada por la lectura del sensor, la integral es la acumulación de áreas que está calculada en base a la fórmula del triángulo (base por altura/2) en un dt dado en milisegundos, tomando en cuenta que este dt es variable y es dado por el muestreo basado en eventos.

```
// Pseudo-código del PID asíncrono

void pid_eventos(void)
{
 error = Setpoint - Sensord;
 proporcional=error;
 integral = integral + ((error+error_anterior)/2.0)*(dt/1000.0);
 derivativo = (error - error_anterior)/(dt/1000.0);
 pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
 error_anterior=error;
}
```

Se debe tener en cuenta que el código anterior no está optimizado. Por ejemplo, algunos coeficientes de la ley de control pueden ser pre calculados para reducir el tiempo de cálculo. Además, el alcance de las variables se considera global. En un enfoque basado en eventos teórico, el detector de evento debe examinar continuamente el sensor y evaluar la condición basada en eventos para activar el generador de eventos y dé paso a la señal del sensor al actuador cuando se convierte en verdad. Como se ha indicado antes, en las implementaciones digitales prácticas el sensor es muestreado con un periodo de muestreo fijo. En función de este periodo, el detector de eventos puede realizar una evaluación discreta o casi continua de la condición basada en eventos (es decir, un enfoque compuesto o híbrido). Sin embargo, una frecuencia de muestreo superior induce mayor costo computacional, además, se tiene que considerar que el periodo de muestreo en el lado del controlador ya no es periódico, pues el muestreo depende de la activación del evento, teniendo en este lado un muestreo basado en eventos en donde el periodo de la señal es estocástico ya que influyen los efectos de la red inalámbrica.

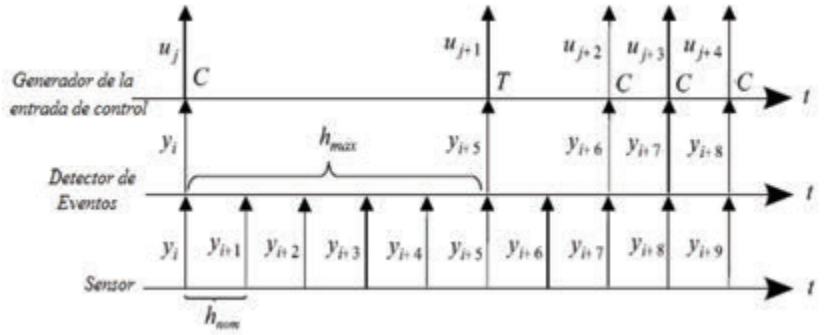
El primer controlador PID basado en eventos reportado en la literatura se describe en la obra de Årzén (1999). El objetivo de este controlador es reducir el tiempo de la CPU para el cálculo de la ley de control, el cual no proporciona una reducción global de la información intercambiada en el lazo de control. En realidad, como ocurre en las realizaciones prácticas de los enfoques basados en eventos, este controlador es una aproximación híbrida entre un tiempo y un controlador basado en eventos. La activación del detector de eventos es la base del tiempo con el periodo de h_{nom}, pero la invocación del generador de entrada de control no lo es. Si se cumple la condición basada en eventos en el detector de eventos entonces se invoca una nueva acción de control dada por el generador de entrada de control. Vale tener en cuenta que si la frecuencia de activación del detector de eventos se aumenta al máximo en una implementación real, este controlador se convierte en un enfoque basado en eventos cuasi-puro, es decir, en teoría continuo.

Como se ha descrito, en cada periodo de muestreo nominal h_{nom}, el controlador lee la muestra del sensor y el detector de evento evalúa la condición basada en eventos con esa información. Si la condición es verdadera, se produce un evento y el generador de entrada de control calculará la acción de control usando una ley PID con muestreo aleatorio de la variable de entrada. La condición lógica es una función del error relativo, por lo tanto, la condición es verdadera cuando el valor absoluto de la diferencia entre el valor actual del error $e(t_{act})$ y el valor del error anterior $e(t_{last})$ es mayor que un umbral predefinido Δ_e. Por lo tanto, un error de muestreo basado en *send-on-delta* se aplica al controlador. Además, el detector de evento incluye una condición de seguridad basado en el tiempo para garantizar un intervalo de tiempo máximo entre eventos consecutivos h_{max}. De esta manera, un nuevo valor de la señal de control se ve obligado a ser calculado cuando el tiempo transcurrido $h_{act} = t_{act} - t_{last}$ excede un límite dado h_{max}. Formalmente, el estado general basado en eventos se puede escribir como:

$$(|e(t_{act})\text{-}e(t_{last})| \geq \Delta e) \quad or \quad (h_{act} \geq h_{max}) \tag{2.3}$$

El patrón de muestreo que produce la condición basada en eventos se denota como muestreo localmente no uniforme (Heemels *et al.*, 2008). La condición basada en eventos utiliza un tiempo de muestreo h_{max} uniforme cuando el error no cambia u oscila sin problemas, pero utiliza un periodo de muestreo h_{act} no uniforme si la variación de error actual se mantiene localmente debajo de Δe antes de que el temporizador fijado a h_{max} se dispare. Cabe señalar que el periodo de muestreo no uniforme h_{act} es siempre un múltiplo de h_{nom}. La figura 13 presenta un ejemplo de la información intercambiada entre los agentes de control utilizando la condición basada en eventos definida en la ecuación (2.3). El generador de entrada de control se ejecuta solo si se detecta el evento, lo que ahorra recursos de cálculo si no hay necesidad de actualizar la variable de control (por ejemplo, cuando el proceso está en el estado estacionario), en este caso estaríamos hablando de un PID asíncrono, pero si el tiempo cuando ocurre un evento aumenta y el generador de eventos no envía información, el controlador deberá estimar una salida en base a un predictor que obedezca a la tendencia o a la función de estado de la planta.

Figura 13
Intercambio de información en el enfoque PID basado en eventos



Fuente: los autores

Alternativas para el controlador PID basado en eventos

El algoritmo PID anterior basado en el enfoque híbrido en donde el detector de eventos actúa periódicamente y en base a una condición lógica o tiempo, dispara el dato al generador de la entrada de control en el cual se realiza el cálculo del algoritmo PID. En este caso se dice que es un PID asíncrono debido a que el dato de salida del PID solo se calcula cuando se activa la condición lógica. En nuestro trabajo también se han implementado otras alternativas como:

- El PID semi síncrono, en donde para el cálculo de la salida del PID, se periodiza el intervalo de tiempo en que llega una nueva actualización de datos enviada por el detector de eventos, es decir, el intervalo de tiempo entre eventos que llega al generador de la entrada de control se divide en un delta de tiempo entero en el cual se calcula el PID, pero existirán intervalos de tiempo en los que no siempre se dará un delta entero, es entonces en donde el PID generará su salida en base a la lectura actual de la entrada proporcionada por el detector de eventos. En donde el *t_* semi síncrono es calculado en base a contadores, iniciando un contador cuando existe una transmisión de datos y actualizando el contador cada vez que hay una transmisión, ese lapso entre contadores se periodiza en base a un determinado número de conteo.

```
// Cálculo de la función PID Semi Síncrono
void pid_eventos(void){
 error = SetPonit - Sensord;
 proporcional=error;
 integral = integral + ((error+error_anterior)/2.0)*(t_semisincrono/1000.0);
 derivativo = (error - error_anterior)/(t_semisincrono/1000.0);
 pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
 error_anterior=error;
t_semisincrono=0;
}
```

- El PID síncrono o periódico que en esencia es similar al semi síncrono con la diferencia de que el cálculo de la salida del PID siempre se dará en base a un periodo delta seleccionado sin importar el tiempo de llegada del nuevo dato enviado por el generador de función, cuya información sirve para actualizar el dato para el cálculo del PID. En este caso se tomó como tiempo de periodo fijo 30 ms.

```
// Cálculo de la función PID Sincrono
void pid_eventos(void){
error = SetPoint - Sensor; //
proporcional=error;
integral = integral + ((error+error_anterior)/2.0)*(0.03);
derivativo = (error - error_anterior)/(0.03);
pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
error_anterior=error;
}
```

En la literatura también existen otras alternativas que pretenden minimizar el cálculo computacional. El uso de un estimador en función a las variables de estado de la planta realizando cálculos que predigan el comportamiento del PID, es un medio eficaz de reducir el coste computacional. Sin embargo, algunas mejoras del algoritmo PID se describen en Durand (*et al.*, 2010) para reducir el coste computacional y lograr un rendimiento similar al de un controlador PID convencional disparado por tiempo.

La minimización del coste computacional implica minimizar el número de eventos activados. La solución propuesta en Durand (*et al.*, 2010) se basa en el cálculo de la señal de control solo cuando la medición es demasiado lejos del valor de referencia. En la concepción original, la condición basada en eventos se basa en el error relativo y utiliza un detector de evento de disparo con h_{nom}. Ahora, la condición lógica del detector se basa en el error absoluto.

$$|e(t_{act})| \geq \Delta e \tag{2.4}$$

Para determinar cuando ocurra el evento, se propone sustituir el detector disparador de evento por uno continuo para enviar una petición en el momento en que el error supera el umbral. En implementaciones reales, el detector de evento de tiempo continuo se emplea mediante la reducción del valor de h_{nom} (el enfoque hibrido) tanto como sea posible, para simular un muestreo continuo.

La condición cuando el número de eventos se reduce al mínimo, sucede en los intervalos de estado estable, es decir, cuando el error permanece por debajo del umbral. Sin embargo, durante el periodo transitorio el número de eventos crecerá y para reducir el número de eventos se introduce una nueva condición lógica que se compone de la adición de un tiempo mínimo entre eventos, es decir, $h_{act} > h_{min}$, al detector de evento. De acuerdo con Cervin y Henningsson (2008), este intervalo de tiempo mínimo podría ser elegido como el periodo de muestreo discreto correspondiente a un controlador de tiempo por alarma. Otra opción es encontrar un compromiso entre la reducción de h_{nom} y el número de eventos producidos. Otra de las mejoras en esta versión de bajo cálculo del algoritmo basado en eventos, es la reducción del error absoluto durante los intervalos de estado estable. Con la implementación actual, $e(t)$ permanece por debajo de Δe, no produce acciones de control para reducir el error.

En los pseudo códigos anteriores se puede apreciar las funciones del PID semi síncrono y síncrono, los cuales se diferencian según la periodicidad o el tiempo con el cual se realiza el cálculo del controlador en el estado estacionario y el transitorio. Simulaciones (Cervin y Åström, 2007) muestran que el comportamiento de este controlador es mejor que el de los otros descritos en esta sección, con un rendimiento muy cercano de un homólogo PID disparado por tiempo utilizando el mismo conjunto de parámetros.

El objetivo del control basado en eventos

A continuación se sintetiza el modelo basado en eventos presentado por Lunze (2014), donde se introduce un enfoque de estado realimen-

tado para el control basado en eventos con el objetivo de diseñar e imitar el comportamiento de un sistema de lazo cerrado continuo dado. En el lazo de control basado en eventos —como se indica en la figura 11— el controlador está incorporado en el generador de entrada de control.

La planta es representada por el modelo de espacio de estado lineal:

$$\dot{x}(t) = Ax(t) + Bu(t) + Ed(t)\,; \quad x(0) = xo \tag{2.5}$$

$$y(t) = Cx(t) \tag{2.6}$$

Donde $x \in \mathbb{R}^n$ denota el estado del sistema con el valor inicial $x_o, u \in \mathbb{R}^m; y \in \mathbb{R}^r$ y son las entradas o salidas medidas, respectivamente, y $d \in \mathbb{R}^t$ representa las perturbaciones externas. El par (A, B) se asume controlable y el límite de la perturbación $d(t)$ será limitado como sigue:

$$\|d(t) \leq d_{max}\| \tag{2.7}$$

La notación $||x||\,y\,||A||$ denota un vector normal arbitrario o inducido de la matriz normal, y el valor absoluto es denotado por $|x|$. La expresión $||x(t)||$ denota un vector normal al tiempo t si el futuro asumimos que:

- La dinámica de la planta se conoce con exactitud.
- El estado $x(t)$ es medible.
- El intercambio de información entre el detector de eventos y el generador de entrada de control es instantáneo y no impone restricciones a la información que debe transmitirse a los tiempos de eventos.

Por lo tanto, la razón para comunicar información a través de las flechas de trazos en la figura 11 se da principalmente por la situación de que la perturbación $d(t)$ ha provocado un comportamiento intolerable de la salida de control $y(t)$ o el estado de la planta $x(t)$.

Como característica principal del esquema propuesto, el detector de eventos utiliza un modelo del circuito de regulación continua para comparar los estados actuales de la planta $x(t)$ con el estado deseado que se produce en el sistema de lazo cerrado continuo. Si la diferencia entre los dos estados excede el límite superior $\bar{e}$, se activa un evento y actualiza el estado $x(t_k)$ para ser transmitido al generador de entrada de control. Es decir, en el lado de la planta cuando no hay transmisión el sistema opera como si estuviera en lazo abierto, pero para que no se desestabilice el sistema debe existir un predictor o estimador del controlador el cual genera la señal para comparar dando un error de estimación que es el cual permite activar el generador de eventos.

Como hecho importante adicional: el generador de entrada de control incorpora una función que depende del proceso para determinar el futuro de la entrada de control $u(t)$, $(t \geq t_k)$ o también se puede interpretar como que en el lado del controlador también existe un predictor o estimador, pero en este caso será de la función de trasferencia de la planta, para que cuando no exista comunicación la salida del generador de entrada de control calcule la señal para enviar al actuador con la señal estimada del comportamiento de la planta. Se muestra que el lazo de control basado en eventos de estas características tiene las siguientes propiedades:

- El estado $x(t)$ del lazo de realimentación de estado basado en eventos es finalmente acotado en el sentido de que sigue siendo, para todos los tiempos t, en una zona acotada Ω_e del estado deseado $x_{CT}(t)$ del lazo de realimentación de estado continuo.
- La comunicación a través del canal de realimentación en el lazo de control basado en eventos es limitada y depende explícitamente de la perturbación $d(t)$.
- Tanto la exactitud de la aproximación del comportamiento del lazo de realimentación de estado continuo como el intervalo de tiempo mínimo entre dos eventos consecutivos (mínimo tiempo entre eventos), se puede ajustar cambiando el e umbral del detec-

tor de eventos con el fin de adaptar el estado de retroalimentación del lazo basado en eventos a las necesidades requeridas.

Realimentación de estado continuo

Las principales propiedades del lazo de realimentación de estado continuo que se utiliza como el sistema de referencia para evaluar el comportamiento del lazo de realimentación de estado basado en eventos. La planta (2.5), (2.6), junto con la retroalimentación de estado:

$$u(t) = -Kx(t) \tag{2.8}$$

Proporcionan el sistema de circuito cerrado continuo:

$$\dot{x}_{CT}(t) = (A\text{-}BK)x_{CT}(t) + Ed(t) \quad ; x_{CT}(0) = xo \tag{2.9}$$

$$y_{CT}(t) = Cx_{CT}(t) \tag{2.10}$$

El subíndice $_{CT}$ se utiliza para distinguir las señales continuas de este modelo de las señales correspondientes del lazo de control basado en eventos considerado más adelante. Se supone que la matriz de realimentación de estado K se ha diseñado de manera que la matriz $\bar{A}$ es Hurwitz y el sistema de lazo cerrado tiene propiedades de atenuación y de perturbaciones deseadas.

Como $\bar{A}$ es Hurwitz y la perturbación $d(t)$ se supone que está limitada de acuerdo con la ecuación (2.7), el estado del lazo de realimentación de estado continuo (2.9), (2.10) es GUUB de acuerdo con la siguiente definición: La solución de $x(t)$ del lazo de control continuo (2.9), (2.10) se dice que es globalmente uniforme en última instancia delimitada (GUUB) si para cada $x_0 \in \mathbb{R}^n$ existe una constante positiva p y un tiempo t tal que mantenga:

$$x(t) \in \Omega_t = \{x : ||x|| \leq p\}, \qquad \forall t \geq \bar{t}$$

Entonces se dice que el lazo de control continuo (2.9), (2.10) está en última instancia, acotado. Para el lazo de control lineal continua (2.9), (2.10) el estado $x(t)$ es GUUB si la matriz $\bar{A}$ es de Hurwitz y la perturbación $d(t)$ es acotada.

Comportamiento del estado-realimentado de lazo continúo

La entrada de control generada por el controlador de estado-realimentado (2.5) está dada por:

$$u(t) = -Ke^{\bar{A}t}x_0 - \int_0^t Ke^{\bar{A}(t-\tau)}Ed(\tau)d\tau \tag{2.11}$$

Esta ecuación muestra que la entrada $u(t)$ no solo depende del estado inicial x_0, sino también de la entrada de perturbación $d(t)$. En el ajuste de control basado en eventos, este aspecto es importante. Si en el momento t_k el estado $x(t_K)$ es comunicado al generador de entrada de control, el generador de entrada de control es capaz de determinar la misma entrada de control $u(t_K) = -Kx(t_K)$ como un controlador de estado de retroalimentación continua. Sin embargo, para todos los futuros tiempos $t \geq t_k$ el generador de entrada de control tiene que saber la perturbación $d(t)$ para $t \geq t_k$, así:

$$u(t) = -Ke^{\bar{A}(t-t_k)}x_{CT}(t_k) - \int_{t_k}^{t} Ke^{\bar{A}(t-\tau)}Ed(\tau)d\tau \quad , \quad t \geq t_k \tag{2.12}$$

Este análisis pone de manifiesto dos hechos importantes:

4. El control de realimentación de estado continuo (2.8) obtiene la información sobre la perturbación actual implícitamente por la comunicación continua del estado actual $x_{CT}(t)$.
5. Cualquier regeneración sin la comunicación continua tiene que hacer suposiciones acerca de la perturbación a ser atenuada. A menos que la alteración sea medible, cualquier regeneración dis-

continua no puede tener el mismo rendimiento que el circuito de retroalimentación con la comunicación continua.

La idea principal del enfoque del estado de retroalimentación basado en eventos discute que lo siguiente es reemplazar la realimentación de estado continuo (2.8) por un controlador basado en eventos, de tal manera que el estado $x(t)$ del lazo de realimentación de estado basado en eventos, para los tiempos t, en el conjunto $\Omega e(x_{CT}(t))$ del estado deseado $x_{CT}(t)$ del lazo continuo de realimentación de estado sea dado por las ecuaciones (2.9) y (2.10).

Realimentación de estados basado en eventos

Generador de entrada de control

Una consecuencia directa del análisis en la sección anterior es que por el momento $t = t_k$ la planta (2.5), (2.6) con la entrada de control (2.8) se comporta exactamente como el lazo de control continua (2.9), (2.10). Si el generador de entrada de control utiliza la ecuación (2.12) para determinar la entrada de control para $t = t_k$, entonces se obtiene el mejor rendimiento posible. Para activar el generador de entrada de control y para utilizar esta ecuación, el estado tiene que ser medido y comunicado al generador de entrada de control, y se tiene que realizar un supuesto acerca de la perturbación.

En lo que sigue, el generador de entrada de control asume que la perturbación es constante: $d(t) = \widehat{d_K}$ *para* $t \geq t_K$ cuando se conoce la magnitud $\widehat{d_K}$. Por lo tanto, se utiliza la ecuación:

$$u(t) = -Ke^{\bar{A}(t-t_k)}x(t_k) - K\bar{A}^{-1}\left(e^{\bar{A}(t-\tau)} - I_n\right)E\,\hat{d}_k, \quad t \geq t_k \qquad (2.13)$$

La cual sigue directamente de la ecuación (2.12) para las perturbaciones constantes, hasta que se obtiene la información siguiente $x(tk + 1)$, donde I_n denota la matriz identidad de tamaño n.

El generador de entrada de control determina la entrada (2.13) por medio de un modelo del sistema de lazo cerrado continuo (2.8):

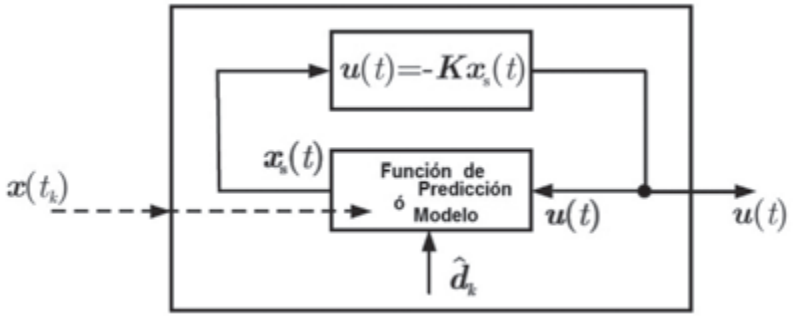
$$\dot{x}_s(t) = \bar{A}x_s(t) + E\hat{d}_k(t) \quad x_s(t_k{}^+) \quad , t \geq t_k \tag{2.14}$$

$$u(t) = -Kx_s(t) \tag{2.15}$$

Aquí, x_s se utiliza para denotar el estado del generador de entrada de control. Se debe tener en cuenta que la señal $u(t)$ obtenida por la ecuación (2.13) es la misma que la solución de (2.14), (2.15).

El tiempo t_k+ indica la actualización del modelo de estados x_s con las medidas de estado $x(t_k)$, que el generador de entrada de control recibe del detector de eventos en el tiempo t_k del evento. La figura 14 muestra el diagrama de bloques del generador de entrada de control.

Figura 14
Generador de la entrada de control



Fuente: los autores

Comportamiento del estado del lazo de realimentación basado en eventos

El análisis de este párrafo es válido para los generadores de eventos arbitrarios y métodos arbitrarios para estimar la magnitud de la perturbación d_k. Se investiga el comportamiento del lazo de control basado

en eventos en el intervalo de tiempo $[t_k, t_k + 1)$ entre los consecutivos tiempos de eventos t_k y $t_k + 1$.

La planta (2.5), (2.6), junto con el generador de entrada de control (2.14), (2.15), se describe para el periodo de tiempo $[t_k, t_k + 1)$ por el modelo de espacio de estado:

$$\begin{pmatrix} \dot{x}(t) \\ \dot{x}_s(t) \end{pmatrix} = \begin{pmatrix} A & -BK \\ 0 & \bar{A} \end{pmatrix} \begin{pmatrix} x(t) \\ x_s(t) \end{pmatrix} + \begin{pmatrix} E \\ 0 \end{pmatrix} d(t) + \begin{pmatrix} 0 \\ E \end{pmatrix} \hat{d}_k \tag{2.16}$$

$$\begin{pmatrix} \dot{x}(t_k^+) \\ \dot{x}_s(t_k^+) \end{pmatrix} = \begin{pmatrix} x(t_k) \\ x_s(t_k) \end{pmatrix} \tag{2.17}$$

$$y(t) = \begin{pmatrix} C & 0 \end{pmatrix} \begin{pmatrix} x(t) \\ x_s(t) \end{pmatrix} \tag{2.18}$$

Este modelo tiene en cuenta que el sistema de lazo cerrado está sujeta a la perturbación $d(t)$, mientras que el generador de entrada de control utiliza la constante d_k estimación de perturbación. La expresión $xi(t + k) = xi(t_k)$ se utiliza para indicar explícitamente que el estado respectivo no se cambia en el instante de tiempo correspondiente.

Detector de eventos

Los eventos se generan mediante la comparación de la medida de la trayectoria de estado $x(t)$ con la trayectoria de estado $x_s(t)$, que se producirían en el lazo de realimentación de estado continuo para la perturbación constante $d(t) = d_k$; como el estado $x_s(t)$ es determinado en función de la ecuación (2.14) que representa la señal deseada de referencia, la medida de estado $x(t)$ debe ser mantenida en el entorno:

$$\Omega_s\big(x_s(t)\big) = \{x: \|x\text{-}x_s(t)\| \leq \bar{e}\} \tag{2.19}$$

De este estado con tamaño ajustable $\bar{e}$.

El detector de eventos desencadena un evento siempre que la diferencia entre las medidas del estado de la planta $x(t)$ y la referencia de estado $x_s(t)$ alcanzan el evento umbral $\bar{e}$.

$$\|x\text{-}x_s(t)\| = \bar{e} \tag{2.20}$$

En este momento t, que denota el tiempo t_k, la información de estado $x(t_k)$ se comunica al generador de entrada de control.

Con el fin de evitar una transmisión continua del estado $x_s(t)$ del generador de entrada de control al detector de eventos, una copia del generador de entrada de control está incluida en el detector de eventos, de modo que el generador de eventos puede determinar el estado $x_s(t)$ por medio de la ecuación (2.14).

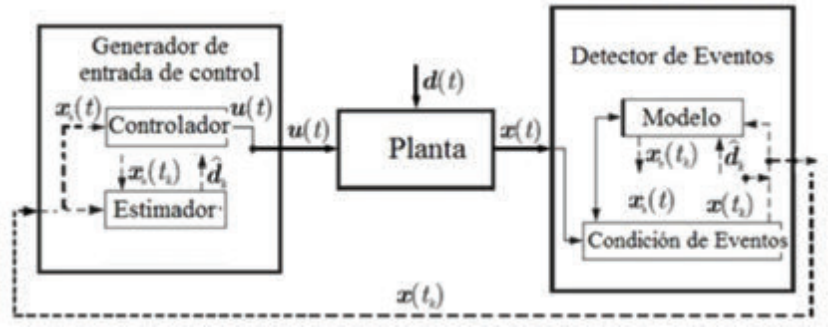
Resumen de los componentes

El lazo de realimentación de estado basado en eventos tiene la estructura representada en la figura 15, y tiene los siguientes componentes:

- La planta (2.5), (2.6).
- El generador de entrada de control (2.14), (2.15).
- El detector de eventos, que incluye una copia del generador de entrada de control (2.14), (2.15) y determina el tiempo del evento t_k de acuerdo con la ecuación (2.20).

En los tiempos de eventos tk, ($k = 0, 1, 2,\ldots$) la información de estado medido $x(t_k)$ se envía desde el detector de eventos hacia el generador de entrada de control y se utiliza allí; así como en el detector de eventos sirve para actualizar el estado del modelo x_s de acuerdo con $x_s(t + k) = x(t_k)$ y determinar la nueva estimación de la perturbación d_k. Dado el supuesto de que la transmisión de datos se lleva a cabo en muy poco tiempo, los modelos del generador de entrada de control y el detector de eventos trabajan sincrónicamente.

Figura 15
Lazo de control de realimentación de estado basado en eventos



Fuente: los autores

Propiedades principales del lazo de realimentación de estados basado en eventos

Las propiedades centrales a investigar al considerar el control basado en eventos conciernen a la estabilidad y la comunicación a través del enlace de retroalimentación. Los principales resultados del análisis posterior son los siguientes:

- El estado $x(t)$ del lazo de control basado en eventos es GUUB y no existe un límite superior en el error de aproximación en términos de emular el comportamiento del lazo de realimentación de estado continuo.
- Existe un límite inferior en el tiempo mínimo entre eventos.
- Si las perturbaciones son suficientemente pequeñas no se genera ningún evento para $t > 0$.

Comparación entre lazo de realimentación de estados basado en eventos y lazo continuo

El siguiente teorema compara el lazo de realimentación de estados basado en eventos con el lazo de realimentación de estado continuo (Lunze y Lehmann, 2010): la diferencia $e(t) = x(t) - x_{CT}(t)$ entre el es-

tado $x(t)$ del lazo de realimentación de estado basado en eventos (2.5), (2.6), (2.14), (2.15), (2.20) y el estado $x_{CT}(t)$ del lazo de realimentación de estados continuo (2.9), (2.10) están acotadas superiormente por:

$$\|e(t)\| \leq e_{max} = \bar{e} \int_0^{\infty} \left\| e^{\bar{A}\tau} BK \right\| d\tau \tag{2.21}$$

Este teorema demuestra que el controlador basado en eventos se puede utilizar para imitar un sistema de retroalimentación de estado continuo con precisión arbitraria, por elegir en consecuencia del umbral $\bar{e}$ de evento. Se puede utilizar para determinar cada límite a la aproximación error $\|e(t)\|$ superior tolerable, el umbral de eventos $\bar{e}$. El precio de una precisión más alta (e_{max} pequeña) es una comunicación más frecuente entre el detector de eventos y el generador de entrada de control. El estado $x(t)$ permanece en el conjunto.

$$x(t) \in \Omega_e\left(x_{CT}(t)\right) = \{x: \|x\text{-}x_{CT}(t)\| \leq e_{max}\} \tag{2.22}$$

Este describe una vecindad limitada del estado $x_{CT}(t)$ del lazo de realimentación de estado continuo para todos los tiempos t, por lo tanto, como el estado $x_{CT}(t)$ del lazo de realimentación de estado continuo es GUUB, el estado $x(t)$ del lazo de realimentación de estado basado en eventos también es GUUB.

El uso de predictores en el control basado en eventos

En los enfoques anteriores, el detector de evento fue considerado disparado por tiempo (con un periodo de muestreo rápido) y el generador de entrada de control basado en evento. Además, se examinó la creación de problemas de tiempo de la naturaleza basada en eventos del generador de entrada de control; estos problemas incluyen periodos variables de muestreo y los grandes errores de aproximación en los términos integral y derivativa. Tales inconvenientes reducen la calidad del control y una heurística debe ser introducida en la ley de control para resolverlos.

Una posible configuración que elimina los problemas de tiempo es que ambos elementos son disparados por tiempo, pero ambos envían la información a su respectivo vecino adyacente cuando se cumple alguna condición basada en eventos. De este modo, el detector de evento evalúa cada h_s su estado basado en eventos (por ejemplo, $|y(t_{act}) - y(t_{last})| \geq \Delta s$), y el generador de entrada de control produce cada h_c una acción de control que se envía al actuador.

Cuando se asume que no hay retrasos en la comunicación y el tiempo de cálculo es insignificante o inferior a los periodos de muestreo h_s y h_c nominales, no hay problemas de tiempo en este enfoque. Sin embargo, el problema con esta configuración es que el generador de entrada de control no podía recibir el estado del proceso del sensor cada h_c, y esta información debe ser obtenida mediante el empleo de un observador (Kalman, Luenberger) en el controlador, para predecir el valor del proceso de variable en cada intervalo de muestreo h_c. En este enfoque, el predictor en realidad se ejecuta en un lazo abierto entre las muestras, y la variable de proceso se calcula utilizando el último estado recibido. La estimación de estado se actualiza cada vez que llega una nueva señal del sensor. Es evidente que la eficacia de la metodología depende en gran medida la exactitud del observador y que la presencia de perturbaciones puede implicar una disminución significativa en el rendimiento. Una solución parcial a estos problemas se puede lograr mediante el empleo de un observador de orden reducido (primer orden) (*cf.* Vasyutynskyy y Kabitzsch, 2009) o mediante una función que describa el proceso, tal como se implementó en las aplicaciones que se describirán más adelante. Otra alternativa, ya en la práctica, es utilizar como estimador la función de la planta como predictor o estimador en el lado del controlador, y en el lado de la planta utilizar un predictor o estimador del controlador en paralelo a la planta.

Un enfoque similar puede ser implementado por la aplicación de un predictor de espacio de estado de primer orden de la planta, para estimar la salida de la planta en un lazo abierto en ausencia de información actualizada desde el detector de evento. El predictor de primer orden simplificado puede ser descrito como:

$$\hat{x}(t) = A_s\hat{x}(t_{k+1}) + B_s u(t_{last}) \tag{2.23}$$

$$y(t_k) = C_s x(t_k) \tag{2.24}$$

Donde A_s, B_s, y C_s son matrices de primer orden y $u(t_{last})$ es la última acción de control enviada al actuador. En este enfoque, el predictor es parte del generador de entrada de control, y por lo tanto se evalúa cada h_c con la información disponible. Al inicio de cada evaluación, se comprueba la llegada de un nuevo valor de la salida de proceso enviada desde el detector de eventos. Si esto ocurre, la ley PID calcula la acción de control. Sí hay llegadas, el controlador utiliza la última predicción del resultado del proceso. Después de eso, el predictor se evalúa con la nueva información y la predicción de salida de proceso está lista para la siguiente invocación del detector de control.

El uso del predictor exige un pequeño h_c para reducir los errores de estimación, aunque esto produce un gran número de acciones de control que pueden saturar el enlace de comunicación o producir desgaste en los actuadores. Para mantener el tráfico de comunicación bajo, este enfoque introduce una condición basada en eventos en el generador de entrada de control, para evaluar si vale la pena transmitir la nueva acción de control al actuador. La condición lógica es:

$$|u(t_{act})\text{-}u(t_{last})| \geq \Delta_c \tag{2.25}$$

Sintonización de los controladores PID

Pese a los avances en la teoría de control en las últimas décadas, los controladores PID ocupan un lugar dominante en la industria. Una encuesta realizada describe que más del 97% de los controladores son PID —descrito por Desborough y Miller (2002)—, por lo que es llamado "el controlador de industrias de proceso por defecto" (Lennartson y Kristiansson, 2009). Esto se debe a que, dada la suficiente experiencia y tiempo, la estructura de un controlador PID es adaptable y fácil de

ser sintonizado, sin la necesidad de una descripción de la planta, solo mediante la observación de la respuesta del sistema. Sin embargo, hay que señalar que los enfoques que ahora se están utilizando del modelo PID son más seguros y más robustos en términos de estabilidad y rendimiento, en una amplia gama de frecuencias.

Además de ser ajustables y poseer gran confianza en la industria, los controladores PID son capaces de satisfacer tanto los requisitos de respuesta transitoria como de estado estacionario de muchos procesos. No hay nada qué ganar con el uso de un controlador más complejo para las plantas cuyas dinámicas dominantes son de segundo orden, que cubre la mayor parte de las plantas industriales.

A continuación presentamos algunos de los métodos mencionados para sintonizar controladores PID, aunque mas detalles al respecto se pueden encontrar en el trabajo de Cominos (*et al.*, 2002).

Clásica sintonización del PID

Consciente de la finalidad de los tres términos, junto con un par de reglas, puede ser adecuado para ajustar el controlador, mediante la observación de la respuesta del sistema. Sin embargo, es difícil juzgar que el controlador PID sintonizado de esa manera es el ideal. Además de esto, aun siguiendo las reglas de oro no siempre son apropiados. En Li (*et al.*, 2006), se ha demostrado que para algunas plantas, aumentando el término derivado se intensifica la ganancia a un nivel en que el sistema se vuelve inestable. Esto es contrario a la creencia general de que la acción derivada aumenta la estabilidad. En el mismo estudio se explica que este tipo de equivocaciones y generalizaciones son la razón para que la mayoría de los controladores PID en la industria tengan parte de la derivación desconectada, por lo que no se utiliza toda la funcionalidad del controlador.

Por estas razones es esencial disponer de una forma sistemática la sintonía de los controladores PID. Afortunadamente, en las últimas décadas numerosos enfoques novedosos para ajustar los parámetros PID

se han propuesto. La mayoría de estos métodos tienen los siguientes objetivos de diseño:

- La estabilidad.
- La atenuación de las perturbaciones de carga, que se puede medir utilizando la integral del error absoluto (IAE).

$$IAE = \int_0^\infty |e(t)|dt \tag{2.26}$$

- La respuesta transitoria que, junto con la IAE, también se puede medir por los siguientes índices de rendimiento:

$$ITAE = \int_0^\infty t|e(t)|dt \tag{2.27}$$

$$ISE = \int_0^\infty e^2(t)dt \tag{2.28}$$

$$ITSE = \int_0^\infty te^2(t)dt \tag{2.29}$$

Método de Ziegler-Nichols

El método de sintonización PID presentado por Ziegler y Nichols (1995) se basa en la respuesta paso de lazo abierto del sistema. Se utiliza el hecho de que muchos sistemas en la industria de procesos pueden ser aproximados por un retardo de primer orden más un retardo de tiempo como:

$$G_p(s) = \frac{\alpha}{sL} e^{-sL} \tag{2.30}$$

Ziegler y Nichols también introdujeron un método basado en la respuesta de frecuencia del sistema de circuito cerrado bajo control proporcional puro. En este caso, la ganancia se incrementa hasta que el

sistema de circuito cerrado se vuelve críticamente estable. En este punto, la ganancia máxima se registra junto con el correspondiente periodo de oscilación, conocido como "periodo final". Sobre la base de estos valores, Ziegler y Nichols calculan los parámetros de ajuste dados en tablas que se indica en Cominos (*et al.*, 2002). Los métodos de ZN fueron diseñados para dar buenas respuestas ante perturbaciones. Un criterio de amplitud de amortiguación se utilizó en el diseño que da un coeficiente de amortiguamiento cerca de 0,2. Esto no es satisfactorio para muchos sistemas, ya que no da márgenes de fase y de ganancia satisfactorios. La exactitud máxima también es grande, debido a que los sistemas son sensibles a las variaciones de los parámetros. Además, los métodos ZN no son fáciles de aplicar en su forma original a las plantas de trabajo.

Asignación de polos

El método analítico de asignación de polos se utiliza sobre todo cuando el sistema en cuestión es de orden inferior. Un enfoque común es adoptar un modelo de segundo orden y luego especificar un factor de amortiguamiento y la frecuencia natural deseada para el sistema. Estas especificaciones, a continuación, se pueden cumplir mediante la localización de los dos polos del sistema en las posiciones que dan los requerimientos de funcionamiento de lazo cerrado. Como un ejemplo, la ecuación característica para un sistema aproximado por un modelo de primer orden es:

$$G_p(s) = \frac{K_p}{1 + sT_1} \tag{2.31}$$

Que bajo el control PI tomará la forma de:

$$s^2 + s\left(\frac{1}{T_1} + \frac{K_p k_p}{T_1}\right) + \frac{K_p k_p}{T_1 T_i} = 0 \tag{2.32}$$

Esto se puede comparar con el modelo general de segundo orden:

$$s^2 + 2\zeta\omega s + \omega^2 = 0 \tag{2.33}$$

Para así obtener:

$$K_p = \frac{2\zeta\omega T_1 - 1}{k} \quad , \quad T_i = \frac{2\zeta\omega T_1 - 1}{\omega^2 T_1} \tag{2.34}$$

En el caso en que se utilice un sistema de segundo orden de la forma siguiente:

$$G_p(s) = \frac{K_p}{(1 + sT_1)(1 + sT_2)} \tag{2.35}$$

Un controlador PID de la forma:

$$G_c(s) = \frac{K_p(1 + sT_i + s^2 T_i T_d)}{sT_i} \tag{2.36}$$

Se pueden colocar arbitrariamente todos los polos en lazo cerrado. Entonces la ecuación característica del sistema se convierte. Esto puede ser comparado con la siguiente ecuación característica de tercer orden en general:

$$(s + \alpha\omega)(s^2 + 2\zeta\omega s + \omega^2) = 0 \tag{2.37}$$

El diseño de polo dominante se basa en el posicionamiento de polos dominantes del sistema en el plano complejo. Por ejemplo, tomando la función de transferencia de un sistema de realimentación unitaria:

$$G_{CL}(s) = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)} \tag{2.38}$$

A continuación, se puede fácilmente encontrar los polos y ceros del sistema de circuito cerrado resultante. En muchos casos, la dinámica del sistema dominante puede ser aproximada por la configuración polo-zero simple. El par de polos *P*1, *P*2 son conocidos como los polos

dominantes. Polos y ceros que tienen partes reales mucho más negativas que las de los polos dominantes, tienen poca influencia en la respuesta general del sistema. Para un controlador PI, como un ejemplo, los polos *P1*, *P*2 pueden ser:

$$P_1 = -\zeta\omega_0 + i\omega\sqrt{1-\zeta^2} \quad , \quad P_2 = -\zeta\omega_0 - i\omega\sqrt{1-\zeta^2} \tag{2.39}$$

Estos dan un sistema con la amortiguación requerida ζ y la velocidad de respuesta dada por ω_0.

Los algoritmos genéticos para la sintonización PID

Los algoritmos genéticos son una zona de rápida expansión en el diseño de sistemas de control. Un algoritmo de ajuste genético comienza generalmente sin conocimiento de la solución correcta y depende de las respuestas de su medio ambiente para dar un resultado aceptable. Se ha demostrado que los algoritmos genéticos son capaces de localizar regiones óptimas en dominios complejos evitando las dificultades, o incluso resultados erróneos en algunos casos, asociados con los métodos de descenso de gradiente y con sistemas de alto orden. Para obtener los parámetros de ajuste del PID, por lo general, se tiene que minimizar un índice de desempeño. Esto, en la mayoría de los casos, es uno de los siguientes: (2.26), (2.27), (2.28) y (2.29).

El algoritmo genético requiere una población de aproximaciones iniciales, que pueden ser al azar, para iniciar la búsqueda. Luego, el algoritmo comprueba la aptitud de cada individuo (o cromosomas). A continuación, se sigue un proceso de selección donde se eligen los individuos más aptos, mientras los individuos restantes son seleccionados probabilísticamente. Los individuos seleccionados se utilizan para producir la siguiente población y luego el proceso se repite hasta que se cumplan los requisitos de diseño. Este método es aplicable a una amplia gama de modelos de sistemas debido a su adaptabilidad. Los sistemas de orden superior no presentan un problema con este procedimiento de ajuste.

Algoritmo heurístico para la sintonización del PID basado en eventos en un WNCS

Los algoritmos heurísticos se caracterizan por ser métodos que buscan soluciones a funciones o a problemas de optimización. En la actualidad, los algoritmos heurísticos han incrementado considerablemente su aplicación, gracias a que encuentran soluciones satisfactorias a problemas complejos, a pesar de no ser capaces de probar que la solución encontrada es la óptima. A continuación, se presentan dos métodos heurísticos: la optimización por cúmulo de partículas (PSO) y los algoritmos genéticos (AG).

Optimización mediante el cúmulo de partículas (PSO)

La meta-heurística fue creada por Eberhart y Kennedy (1995) como un algoritmo estocástico. El PSO fue originalmente desarrollado para la optimización en espacios continuos, pero también ha sido desarrollado para espacios discretos, presentando un buen rendimiento cuando se aplica a funciones objetivas discontinuas, por lo que es utilizado en la optimización en muchos campos de la ingeniería. El PSO simula el vuelo inteligente de un cumulo de aves, se ha caracterizado por una optimización global y una rápida convergencia. En nuestro proyecto se utiliza el algoritmo PSO para sintonizar los parámetros del controlador PID.

Mediante el algoritmo PSO se realiza la sintonización y se encuentran los parámetros óptimos del controlador PID, como las constantes K_p, K_i, K_d. Se implementa un controlador PID digital que viene de la discretización de un PID análogo y obedece a la relación: $u(k) = K_p e(k) + K_I \sum_{j=0}^{k} e(j) + K_D[e(k)\text{-}e(k-1)]$, donde K_p es el coeficiente proporcional, K_i es el coeficiente integral, K_d es el coeficiente diferencial, $u(k)$ es la salida de control en un tiempo de muestreo k el cual es variable, pero para efectos de la simulación se considera un sistema continuo periódico.

Parámetros del algoritmo PSO

El PSO se caracteriza por ser un algoritmo iterativo y estocástico sobre un enjambre de individuos denominados "partículas inteligentes". La posición de cada partícula representa una solución potencial al problema de optimización. Una partícula está compuesta por tres vectores:

- El vector de posición $x_i = [x_{i1}, x_{i2}, \ldots\ldots, x_{in}]$ el cual guarda la posición actual de la partícula que se toma como referencia para calcular la nueva dirección de búsqueda.
- El vector $pBest_i$ ó $pBest_i = [p_{i1}, p_{i2}, \ldots\ldots, p_{in}]$ el cual guarda la posición de la mejor solución encontrada hasta esa generación y representa la memoria de la partícula.
- El vector de velocidad $v_i = [v_{i1}, v_{i2}, \ldots\ldots, v_{in}]$ el cual guarda el gradiente (dirección del vuelo de la partícula).

El flujo del algoritmo PSO se determina con el siguiente procedimiento:

1. Se inicializa la población de las partículas, incluyendo los vectores de posición y velocidad aleatoriamente.
2. Se evalúa la función objetivo para cada partícula.
3. Se compara la función objetivo de cada partícula con la posición óptima local, si es la mejor, se guarda la posición de la partícula.
4. Se compara la función objetivo de cada partícula con la posición global, si es la mejor, se guarda.
5. Se actualiza la velocidad y posición de la partícula dados en los pasos 3 y 4.
6. Si la condición de finalización termina, entonces se evalúa otra partícula (paso 2).

El modelo matemático que representa el algoritmo PSO se presenta en la ecuación (2.40), que modifica la velocidad de la partícula o la dirección de búsqueda, y en la ecuación (2.41), que modifica la posición de la partícula.

$$v_i^{k+1} = wv_i^k + c1\ rand_1\left(pBest_i\text{-}x_i^k\right) + c2rand_2\left(gBest_i\text{-}x_i^k\right) \quad (2.40)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (2.41)$$

En donde: $i = 1, 2,\ldots m$, siendo m el número total de partículas de la población; *Vi* es la velocidad de la partícula; *pBest* es la posición optima de la partícula; *rand*() es un número randómico entre 0 y 1; X_i es la posición de la partícula; $c1$,$c2$ y w son factores de aprendizaje, siendo $c1$ una componente cognitiva, $c2$ el componente social y w el factor de inercia.

Antes de comenzar con el algoritmo PSO se define el número de partículas que harán la búsqueda. Para esta variable no existe en la literatura un número recomendable, pero como es de suponer, si se ocupa una cantidad enorme de partículas eso implicaría que el espacio de búsqueda sea igualmente enorme; de tal manera que se tendrá una mayor probabilidad de encontrar la solución en pocas iteraciones. Tener un número cuantioso de partículas puede resultar contraproducente debido a que cuantificar la aptitud de cada partícula hará que el tiempo de búsqueda se incremente considerablemente. Por otra parte, el número de partículas también pude estar limitado por la memoria de los dispositivos electrónicos.

En nuestro trabajo se eligió aves que buscaran la solución con un número máximo de iteraciones. Estos números se establecieron de acuerdo a pruebas hechas en simulaciones. Cabe mencionar que la dimensión del problema es de orden tres, debido a que se cuenta con tres variables: K_p, K_i, K_d.

En la etapa de inicialización de las variables también se definen los parámetros del PSO, en este caso: $c1 = 0{,}2$ y $c2 = 0{,}2$ los cuales son la razón o rapidez de aprendizaje del componente cognitivo y social, respectivamente. También se inicializó los parámetros R1 y R2 de manera aleatoria, en un intervalo de (0 1). Posteriormente, se inicializó la posición y la velocidad de cada partícula de manera aleatoria.

Una vez definidos los parámetros del PSO y generados las partículas, se cuantifica la aptitud de cada partícula. La cuantificación de la aptitud de cada partícula se hace a través de la integral del error al

cuadrado o como se conoce comúnmente: ISE (por sus siglas en inglés: Integral Squared Error). La intensión de este método es minimizar el error de salida en la función del tiempo transcurrido.

$$\mathrm{ISE} = \int_0^{\tau} \mathrm{e}^2(\mathrm{t})\mathrm{dt} \tag{2.42}$$

El ISE es relativamente insensible a pequeños errores, pero los grandes errores contribuyen fuertemente al valor de la integral. Consecuentemente, utilizar el ISE como criterio de optimización dará como resultado una respuesta con pequeños sobrepasos en el sistema, pero con un largo tiempo de estabilización, puesto que los pequeños errores a lo largo del tiempo contribuyen muy poco a la integral.

Existen otros criterios de optimización, como utilizar la integral del valor absoluto del error IAE (por sus siglas en inglés: Integral Absolute Error). Este criterio es más sensible a pequeños errores, pero es menos sensible que el ISE a grandes errores:

$$IAE = \int_0^{\tau} |e(t)| dt \tag{2.43}$$

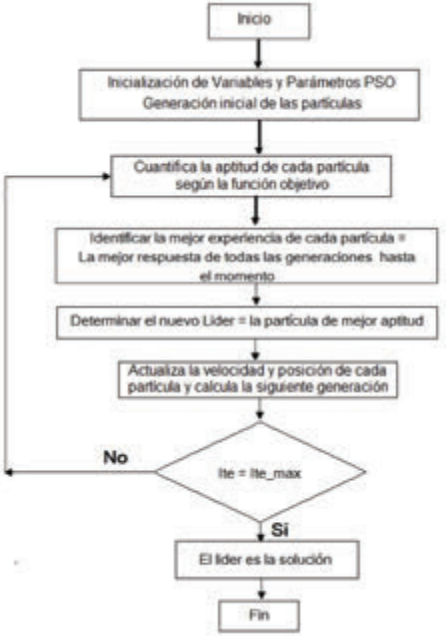
También se encuentra la integral del tiempo multiplicado por el valor absoluto del error ITAE (por sus siglas en inglés: Integral Time Absolute Error). El ITAE es insensible a los errores iniciales —y a veces inevitables—, pero penaliza fuertemente los errores que permanecen a lo largo del tiempo. La respuesta óptima definida por ITAE, consecuentemente, mostrará tiempos cortos en la estabilidad y un mayor sobrepaso que los otros criterios en la respuesta del sistema:

$$ITAE = \int_0^{\tau} t|e(t)| dt \tag{2.44}$$

Una vez finalizado el proceso de cuantificar la aptitud de cada individuo o partícula, se procede a buscar el individuo con la mejor aptitud, en otras palabras: como se desea encontrar el mínimo del siste-

ma, entonces se busca el individuo que presenta un número menor de ISE. Este individuo será el líder que guiará al resto de la población. En la primera generación, cada individuo no cuenta con un historial, de tal manera que su experiencia es su posición actual. De esta forma se determina la siguiente posición y velocidad, de acuerdo a las ecuaciones (2.40) y (2.41), las que generan la nueva generación. Por ende, se repite el procedimiento hasta cumplir con el número máximo de iteraciones. Cabe mencionar que si en una de las generaciones existe un individuo que exhibe una mejor aptitud que el líder actual, el individuo con la mejor aptitud será el nuevo líder que guiará al resto de la población. De igual manera, la experiencia de cada individuo va evolucionando. Por último, al terminar el proceso, el líder será la solución del problema.

Figura 16
Diagrama de flujo para el algoritmo PSO



Fuente: los autores

Algoritmo PSO para sintonizar un controlador PID

1. Inicializa variables, donde: n = tamaño del cúmulo; *n_bird* = número de iteraciones; *dim* = dimensión del problema; $c1 = c2$ = razón de aprendizaje; w = factor de inercia. Y se inicializan los parámetros y rangos de búsqueda.
2. Generación de una población inicial.
3. Evaluación inicial de la población.
4. Selección de la mejor aptitud en base a una función objetivo.
5. Cálculo de la velocidad, posición.
6. Evaluación de una nueva partícula. (Condición para finalizar la búsqueda).

```
For ite = 1 : n_bird %
```

7. Evaluación de la aptitud de cada partícula.

```
For i = 1 : n
aptitud_actual(i) = fModeloBB(pos_actual(:,i));
end
```

8. Encontrar la mejor aptitud local.

```
for i = 1 : n
        if aptitud_actual(i) < aptitud_local_best(i)
        aptitud_local_best(i) = aptitud_actual(i);
        pos_local_best(:,i) = pos_actual(:,i);
        end
end
```

9. Encontrar la partícula con la mejor aptitud actual.

```
[aptitud_actual_global_best, g] = min(aptitud_local_best);
Ap_act_glob_beite(ite)=aptitud_actual_global_best;
```

10. Cambiar el líder o seguir con el mismo según su aptitud.

```
If aptitud_actual_global_best < aptitud_global_best
aptitud_global_best = aptitud_actual_global_best;
        for i = 1 : n
        pos_global_best(:,i) = pos_local_best(:,g);
        end
end
```

11. Calcular la velocidad.

```
W*velocidad + c1*(R1.*(pos_local_best – pos_actual)) +
c2*(R2.*(pos_global_best – pos_actual));
```

12. Calcular la posición.

```
Pos_actual = pos_actual + velocidad;
```

13. Evolución de las partículas.

```
Kpv1(ite) = pos_global_best(1,1);
kiv1(ite) = pos_global_best(2,1);
kdv1(ite) = pos_global_best(3,1);
end
[Jbest_min,I] = min(aptitud_actual);% Aptitudes mínimas
pos_actual(:,I); % La mejor solución
```

14. Solución encontrada por el PSO al controlador PID.

```
Kp1 = pos_actual(1,I); % Ganancia proporcional
ki1 = pos_actual(2,I); % Ganancia integral
kd1 = pos_actual(3,I); % Ganancia derivativo

k = [kp1 ki1 kd1];
```

Optimización mediante algoritmo genético (AG)

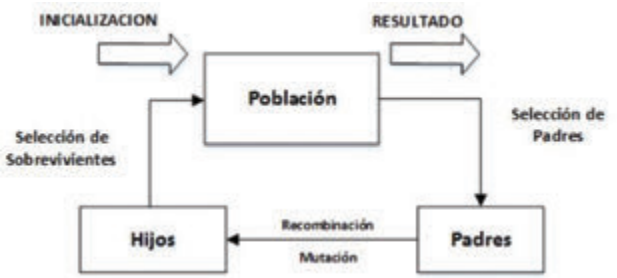
El AG es una técnica de búsqueda que se utiliza para optimizar sistemas o encontrar soluciones a funciones. El AG está inspirado en la evolución de Charles Darwin, es decir, se basa en los mecanismos de selección que utiliza la naturaleza, donde los individuos más aptos de una población son los que sobreviven al adaptarse más fácilmente a los cambios que se producen en su entorno. El principio básico del algoritmo genético fue establecido por John Holland en 1975. De acuerdo a la literatura, existen muchas variantes de los AG, pero la esencia de esta técnica es la misma: dada una población de individuos (posibles soluciones del problema), la influencia del ambiente o el entorno ocasiona selección natural (la supervivencia del más apto), misma que da lugar a un incremento en la aptitud de la población.

En una forma explícita, se diseña una función de calidad a ser maximizada o minimizada, posteriormente se crea de forma aleatoria un conjunto de soluciones candidatas, es decir, elementos del dominio de la función, y se aplica la función de calidad como una medida de aptitud abstracta. En base a la aptitud de cada individuo, algunos de los mejores candidatos son seleccionados para pasar a la siguiente generación y aplicarles recombinación y/o mutación. La recombinación es un operador aplicado a dos o más candidatos seleccionados (llamados "padres"). Realizando la recombinación y la mutación se conduce a un conjunto de nuevos candidatos ("hijos") que compiten junto con los padres por un lugar en la siguiente generación. Este proceso puede ser iterativo hasta que un candidato con suficiente calidad (una solución) sea encontrado o hasta que algún criterio de finalización establecido sea previamente alcanzado (figura 17).

El AG es una herramienta que pertenece a la rama de la inteligencia artificial, la cual ha probado ser útil para encontrar soluciones a problemas reales. Como se ha mencionado, el AG es una herramienta de optimización, un enfoque a problemas de control. La optimización de una función, por lo general, resulta ser el error entre la referencia y la salida del

sistema. En muchos sistemas se espera que la respuesta del sistema tenga tiempos cortos, sobrepasos pequeños y sin errores a lo largo del tiempo.

Figura 17
Algoritmo genético



Fuente: los autores

El AG implementado

1. Primera generación. Como en la mayoría de los programas, el primer paso es inicializar las variables a utilizar, sin embargo, en el AG se genera la primera generación tomando en cuenta el espacio de búsqueda de la solución. Asimismo, el espacio de búsqueda se genera de forma aleatoria y a cada elemento de esta generación se le conoce como "individuo", donde cada uno es una posible solución al problema. Otro factor que se considera al generar la primera generación es el tamaño de la población, es decir, el número de individuos que competirán para sobrevivir, y por último, qué tan aleatorio se pretende generar el espacio de búsqueda.
2. Evaluar aptitud de cada individuo. Para encontrar la solución al problema de optimización, se debe evaluar cada individuo de la generación en una función de su calidad o aptitud. Al someter a cada individuo a la función de aptitud, lo que se hace es cuantificar sus cualidades, y bajo algún criterio se selecciona a los individuos que participarán en la recombinación y generarán nuevos candidatos, los cuales competirán por sobrevivir y ocupar un espacio en

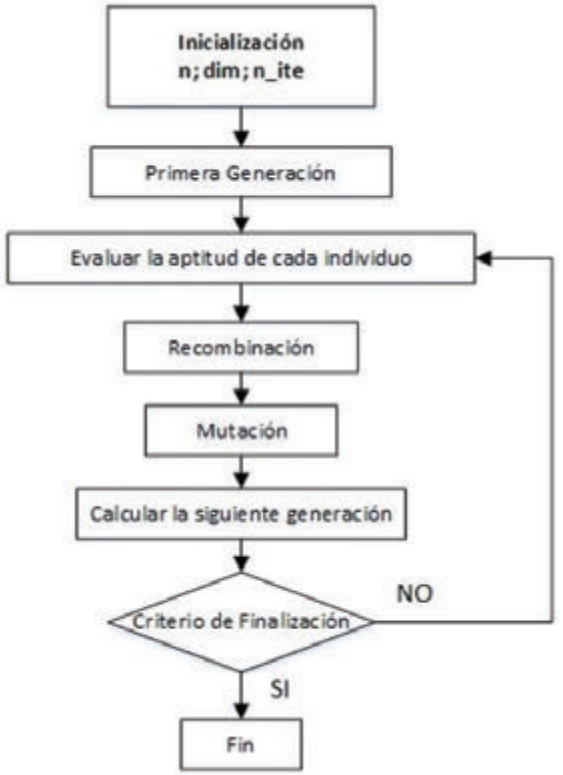
la siguiente generación; incluso puede llegar el caso de que uno de estos elementos cumpla con todas las cualidades buscadas y sea la solución. La función de aptitud es un punto crítico del AG, ya que esta función debe ser capaz de poner de manifiesto la aptitud de cada individuo para el problema que se desea optimizar.

3. Selección. Después de conocer la aptitud de cada uno de los individuos se procede a elegir los que serán padres. Para este proceso existen diferentes técnicas de selección, una de ellas es seleccionar la mitad de la generación que haya mostrado ser la más apta para sobrevivir, a este proceso se le conoce como "elitismo". Sin embargo, esta técnica en algunos problemas presenta convergencia prematura, en cuyo caso existen otras alternativas para hacer la selección de padres, por ejemplo, la selección aleatoria o la conocida como "ruleta".
4. Operador de recombinación. Una vez hecha la selección de los padres se prosigue a hacer el cruce entre padres, el operador que más se utiliza es seleccionar dos padres y a partir de ellos crear dos hijos, la generación de hijos se conoce como el proceso de recombinación. Existen diversas maneras de generar los hijos, dependiendo de la codificación que se usa, en nuestro trabajo usamos números reales y el operador de recombinación se hizo de manera aritmética.
5. Mutación. La mutación es opcional, en muchos de los casos es preferible contar con este operador, ya que hay la posibilidad que el AG llegue a una solución local, donde este operador ayude a encontrar una solución global. Cabe mencionar que la mutación surge de manera aleatoria y uno de los individuos de la población mutará, es decir, cambiará completamente por otro agente del espacio de búsqueda que es generado de forma aleatoria.
6. Generación siguiente. Para generar la siguiente generación se toma a los padres que fueron seleccionados con respecto a su aptitud y los hijos que previamente fueron creados a partir de la recombinación de los padres, además de la mutación si esta fue generada. La nueva generación cuenta con el mismo número de

individuos que la generación pasada y están listos para competir por un puesto en la siguiente generación.

7. Criterio de finalización. Este paso es esencial para que el AG detenga la búsqueda, esta puede ser detenida si uno de los individuos presenta las aptitudes deseadas, es decir que entonces el AG llegó a la solución. O puede detenerse si ya cumplió con un número de generaciones determinado, es decir, un número de iteraciones que previamente se ha elegido.

Figura 18
Diagrama de flujo del algoritmo genético



Fuente: los autores

AG para sintonizar un controlador PID

1. Inicialización: *n* = tamaño de la población; *dim* = dimensión del problema; *n_ite* = número de iteraciones (generaciones).
2. Rangos iniciales para constantes PID.
3. Generación de la población con distribución normal dentro del rango. (Condición para finalizar las iteraciones).

```
For ite = 1 : n_ite %
```

4. Analizar la aptitud de cada uno de los individuos.

```
For i = 1 : n
aptitud_actual(i) = fModeloBB(P(:,i)); % Función para la aptitud.
End
```

5. Ordenar de menor a mayor.

```
[FA ind] = sort(aptitud_actual);
Faite(ite)=FA(1,1);
```

6. Selección de los padres que se reproducirán.

```
For i = 1 : n
        Px(i) = P(1,ind(i));
        Py(i) = P(2,ind(i));
        Pz(i) = P(3,ind(i));
end
```

7. Combinación y generación de los hijos.

```
For i=1:n/4
        Hx1(i)=Px(i);
        Hy1(i)=Py((n/2+1)-i);
        Hz1(i)=Pz((n/2+1)-i);
        Hx2(i)=Px((n/2+1)-i);
        Hy2(i)=Py(i);
        Hz2(i)=Pz(i);
end
```

```
P1 = Px;
P2 = Py;
P3 = Pz;
P1((n/2+1)☹3/4*n))=Hx1;
P2((n/2+1)☹3/4*n))=Hy1;
P3((n/2+1)☹3/4*n))=Hz1;
P1((3/4*n+1):n)=Hx2;
P2((3/4*n+1):n)=Hy2;
P3((3/4*n+1):n)=Hz2;
```

8. Nueva población.

```
P(1,1:n) = P1;
P(2,1:n) = P2;
P(3,1:n) = P3;
```

9. Individuo con mejor aptitud.

```
Kpv1(ite) = P1(1);
kiv1(ite) = P2(1);
kdv1(ite) = P3(1);
end
```

10. Controlador PID

```
kp1 = P1(1) % Ganancia Proporcional
ki1 = P2(1) % Ganancia Integral
kd1 = P3(1) % Ganancia Derivativo
% Controlador PD
k=[kp1 ki1 kd1];
```

11. Gráficos

Simulaciones de los algoritmos heurísticos para sintonización

A continuación se presentan los resultados que se obtuvieron en las simulaciones. Las simulaciones se realizaron en Matlab. Para evaluar

los algoritmos heurísticos se realizó la simulación sobre una función objetivo, que se obtuvo experimentalmente mediante la *toolbox* IDENT de Matlab, en la cual se ingresaban unos valores de referencia y los valores de salida dada por la posición de la bola en la viga mediante el sensor de distancia. La función objetivo evalúa las constantes del PID bajo el criterio de la IAE:

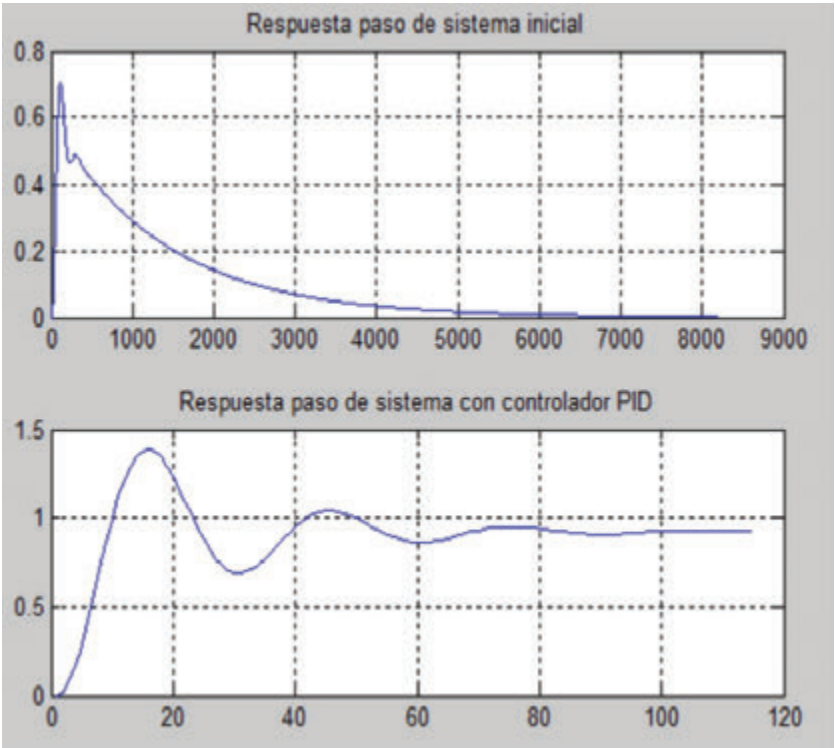
```
function[IAE]=fCtrlPlantaBB(k)
%********** FUNCION OBJETIVO DE UN PROCESO BEAN & BALL
*********
clc;
num=[6.049 0];
den=[2.8 11.05 10.64 4.569 0.2515];
error=0;
dt=0.03; Tmax=5;
t=0:dt:Tmax;
s=tf('s');
Planta=tf(num,den);
sys_c0=feedback(Planta,1);
% Controlador PID con valores iniciales experimentales de:
% kp=8; ki=0.5;kd=2.8;
kp=k(1);ki=k(2); kd=k(3);
C= pid(kp,ki,kd);
% Sistema lazo cerrado realimentación Unitaria
sys_c1=feedback(C*Planta,1);
m= step(sys_c1,t);
% Cálculo del error
error=1-m;
IAE=trapz(t,abs(error));
```

La función de transferencia del sistema está dada por:

$$FT_{sistema} = \frac{6.049s}{2.8s^4 + 11.05s^3 + 10.64s^2 + 4.569s + 0.2515} \quad (2.45)$$

Generando las siguientes respuestas:

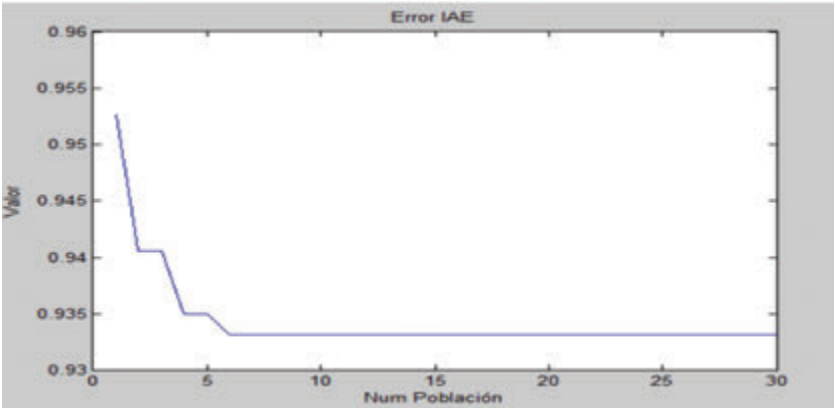
Figura 19
Respuestas del sistema a una señal paso



Fuente: los autores

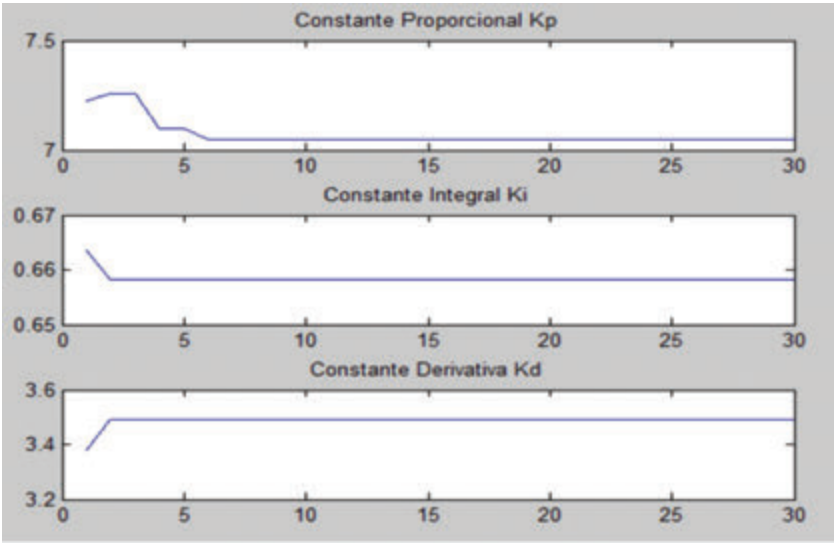
Los resultados generados por el AG dieron los siguientes valores de las constantes PID bajo el criterio de error del IAE, como se muestran en las figuras 20 y 21:

Figura 20
Error generado por el algoritmo genético bajo el criterio IAE



Fuente: los autores

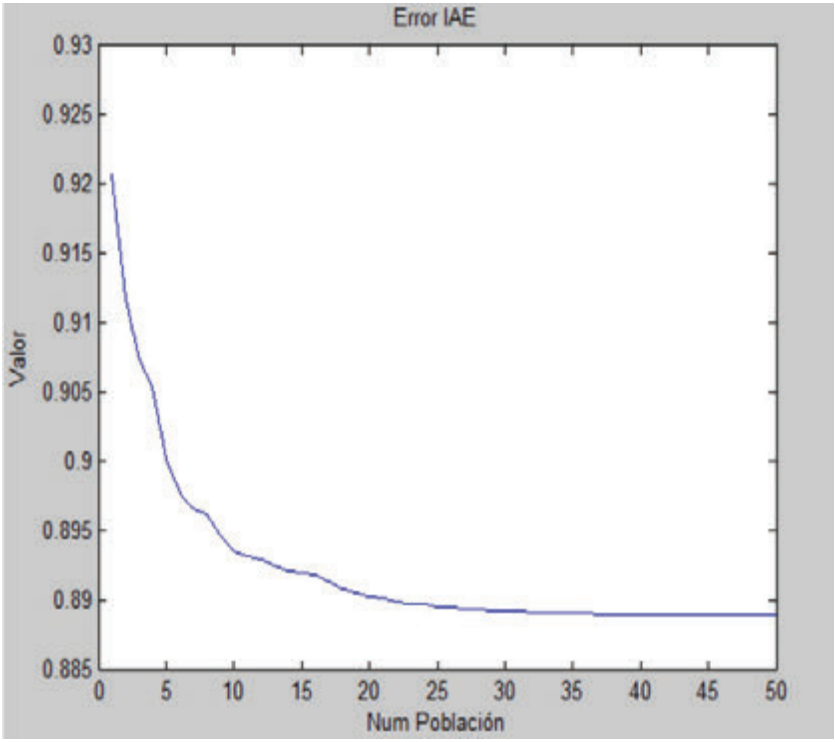
Figura 21
Constantes K_p, K_i, K_d con AG



Fuente: los autores

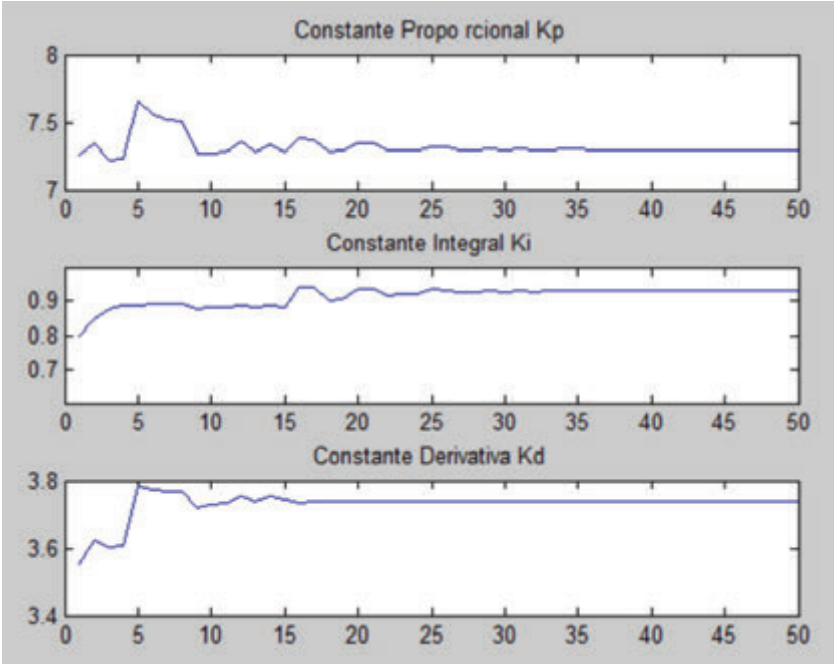
Los resultados generados por el algoritmo PSO dieron los siguientes valores de las constantes PID bajo el criterio de error del IAE, como se muestran en las figuras 22 y 23:

Figura 22
Error generado por el algoritmo PSO bajo el criterio IAE



Fuente: los autores

Figura 23
Constantes K_p, K_i, K_d con algoritmo PSO



Fuente: los autores

En la tabla 2 se muestran los valores obtenidos al utilizar los algoritmos heurísticos y evaluarlos con la función objetivo. En la figura 19 se observa la respuesta de la función de transferencia del sistema obtenida de valores experimentales ante una entrada de tipo escalón, que se refiere a una posición deseada de la bola en la viga. En la figura 19 (superior) se ve que la respuesta de la simulación sin controlador es inestable, mientras en la misma figura (inferior) también se ve el comportamiento de la planta con controlador PID. Además, se puede evidenciar que tiene un sobreimpulso (SO) del 38%, pero la integral del error absoluto (IAE) es demasiado alta: 11,134.

Tabla 2
Datos de los tiempos de retardo en PID asíncrono con criterio IAE

Valores obtenidos al analizar la función de transferencia de la planta obtenida	Valores obtenidos al sintonizar con AG	Valores obtenidos al sintonizar con algoritmo PSO
$K_p = 8$	$K_p = 7{,}04644$	$K_p = 7{,}2954$
$K_i = 0{,}5$	$K_i = 0{,}6582$	$K_i = 0{,}9325$
$K_d = 2{,}8$	$K_d = 3{,}4903$	$K_d = 3{,}7377$
IAE = 11,134	IAE = 0,9331	IAE = 0,8890
Máximo SO = 0,38	Máximo SO = 0,28	Máximo SO = 0,29
Tiempo pico = 12,6	Tiempo pico = 12	Tiempo pico = 11,7
Tiempo de subida = 0,75	Tiempo de subida = 0,72	Tiempo de subida = 0,72
	Tiempo de corrida = 49,79seg	Tiempo de corrida = 52,53seg

Fuente: los autores

Al simular la función de la planta con los parámetros obtenidos luego de sintonizar con las constantes K_p, K_i, K_d dadas por el AG, observamos que disminuye el SO al 28%, pero baja el IAE radicalmente. Lo mismo se tiene cuando se simula el comportamiento de la planta con las constantes obtenidas por el algoritmo PSO.

Los valores K_p, K_i, K_d obtenidos por los algoritmos heurísticos de sintonización están dentro de un rango de búsqueda que fue configurado en base a los siguientes valores obtenidos experimentalmente: $K_p\_ini = 0$; $K_p\_fin=10$; $K_i\_ini = 0$; $K_i\_fin=2$; $K_d\_ini = 0$; $K_d\_fin = 6$. Al simular y correr los algoritmos se obtuvieron valores dentro del rango especificado, es decir, se limitó el campo de búsqueda de la mejor solución. De lo contrario, el espacio de búsqueda puede ser ilimitado, es decir, nos puede dar cualquier valor que cumpla con la función objetivo, considerando que esos valores no siempre pueden implementarse en la realidad y por ende no siempre se puede obtener un control óptimo.

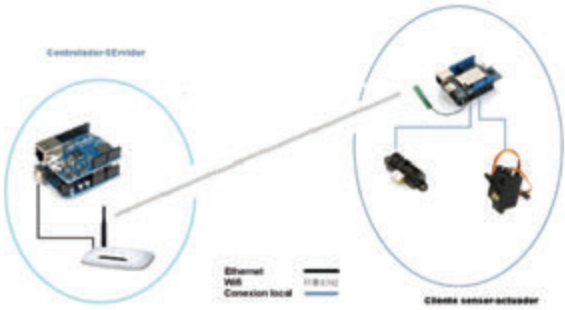
Capítulo tres

Implementación de un WNCS aplicado al controlador PID predictivo basado en eventos para el sistema BB

Descripción de la arquitectura de los dispositivos que conforman el WNCS

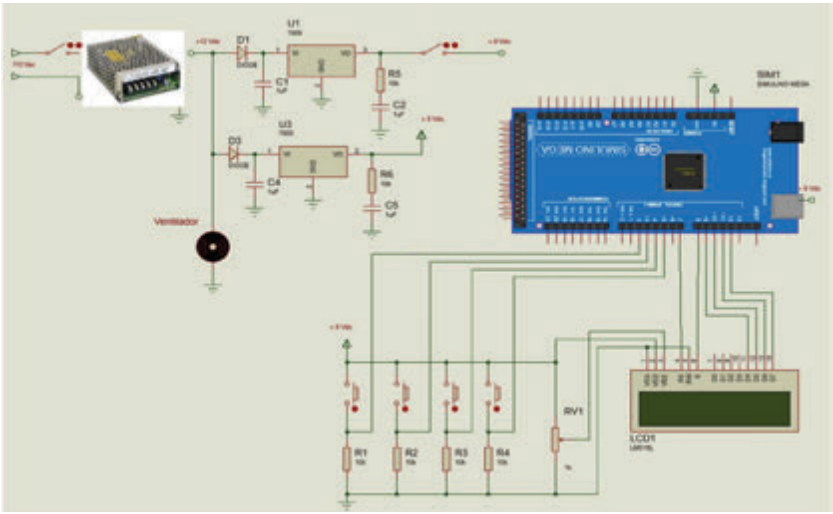
El cliente (sensor-actuador) se halla localizado directamente en la planta, en este nodo se usa Arduino Yun, que permite comunicación por Wi-Fi. En un nodo remoto se halla Arduino Ethernet (controlador), que está conectado a un *router* y trabaja como servidor en la red. Cada dispositivo dentro de la red tiene una dirección IP única que sirve como parámetro de identificación. El servidor tiene una dirección fija a diferencia del cliente que tiene una dirección dinámica que es otorgada por el *router* a través del protocolo DHCP.

Figura 24
Elementos de red



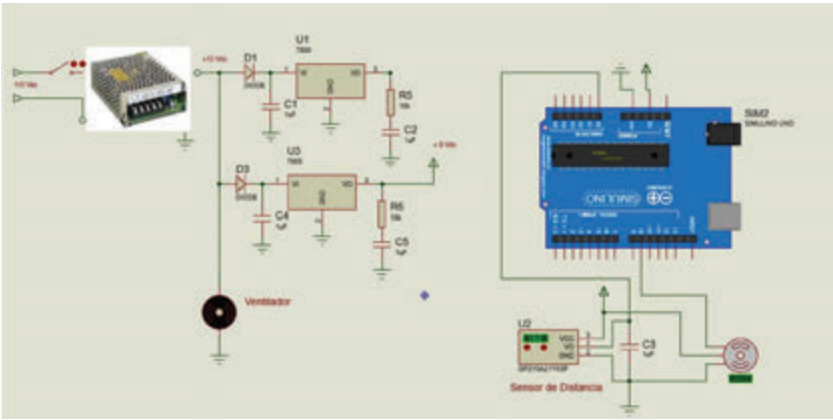
Fuente: los autores

Figura 25
Diagrama esquemático del módulo servidor



Fuente: los autores

Figura 26
Diagrama esquemático del módulo cliente

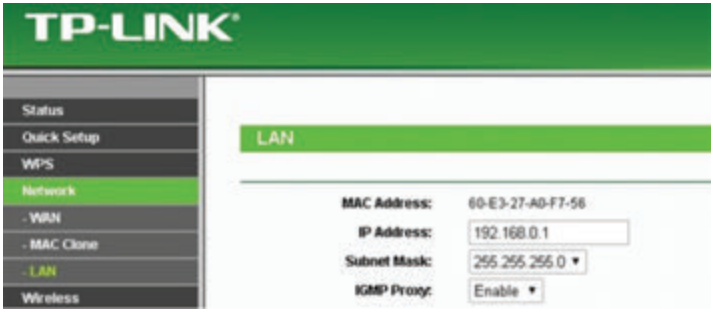


Fuente: los autores

Elementos de red

En el presente proyecto se usa un *router* marca Tp-Link (TL-WR841N) y se configura una red LAN con direcciones 192.168.0.X y sub máscara 255.255.255.0:

Figura 27
Configuración del *router* Tp-Link



Fuente: los autores

Se configura la red externa (WAN). Dependiendo del proveedor de internet tendremos una dirección pública fija o dinámica. En este caso se selecciona una IP fija:

Figura 28
Configuración de red WAN

Fuente: los autores

Se debe de configurar el protocolo DHCP que otorga direcciones IP a los diferentes elementos del sistema. Se define un *pool* de direcciones (192.168.0.100……192.168.0.199):

Figura 29
Configuración del protocolo DHCP

Fuente: los autores

Servidor-controlador

Arduino Ethernet Shield permite a una placa Arduino conectarse a internet. Esta se basa en el chip Wiznet W5100 Ethernet. El W5100 Wiznet proporciona una red IP capaz de usar los protocolos TCP y UDP, y tiene un estándar de conexión RJ-45. Sus características son:

- Tensión de alimentación de 5V.
- Controlador Ethernet: W5100 con una memoria interna de 16Kb.
- Velocidad de trasmisión de 10/100Mb.
- Conexión con Arduino a través del Puerto SPI (más detalles en Gutiérrez, 2010).

En este nodo se usa Arduino Ethernet, que estará conectado directamente al *router*. Por ser un servidor, su dirección IP es fija y no es asignada por el protocolo DHCP. Arduino Ethernet permite configurar desde su programación interna la dirección IP y MAC que se desean usar dentro de la red:

Figura 30
Parámetros de red en Arduino

servifor

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 0, 155);
EthernetServer server(80);
```

Fuente: los autores

Los diferentes algoritmos de control son programados en este nodo. El controlador responde a las diferentes peticiones del cliente (planta) encapsulando el resultado que se obtiene del PID en la respuesta del servidor hacia el cliente.

Cliente sensor-actuador

El Arduino Yun es el primer miembro de una nueva serie de placas Arduino que combinan la potencia de Linux junto con la sencillez característica de Arduino. Combina el chip del modelo Leonardo (ATMega32U4) junto con un módulo SOC (System-On-a-Chip), corriendo una distribución de Linux llamada Linino, basada en OpenWRT. Una de las características más interesantes es que soporta red cableada Ethernet y Wi-Fi. Dispone de dos conexiones de red. Una red Ethernet 10/100 Mbps y otra Wi-Fi (IEEE 802.11 b/g/n, 2,4GHz) que puede montarse como cliente o como punto de acceso.

En la planta se usa Arduino Yun, que está conectado directamente al sensor y actuador, es decir, este nodo recibe la información de la planta en tiempo real. Para configurar Arduino Yun a la red de trabajo, primero se establece comunicación directamente vía Wi-Fi con la red de nombre Dragino-XXX-XXXXX (nombre que viene dado por el fabricante):

Figura 31
Red Wi-Fi de Dragino Yun

Fuente: los autores

En esta red se presenta una aplicación en la dirección IP 192.168.240.1 que permite configurar Arduino Yun a redes que se hallan dentro de su cobertura:

Figura 32
Configuración de Dragino Yun

Fuente: los autores

Se identifica la red de trabajo y se introduce la contraseña. Una vez realizado esto, el *router* otorga una dirección automáticamente a Arduino YUN. La configuración de Arduino Yun como cliente es muy simple, solo se requiere de dos librerías inicialmente configuradas: HttpClient.h —que sirve para configurar el dispositivo como un cliente en la red— y Bridge.h —para comunicar el pequeño ATMega32U4 con el módulo Linux—:

Figura 33
Librerías de Dragino Yun

```
#include <Servo.h>
#include <Bridge.h>
#include <HttpClient.h>
#include <Console.h>
```

Fuente: los autores

Para realizar peticiones al servidor se crea una variable tipo String, que lleva la dirección del servidor y el valor de la variable del sensor:

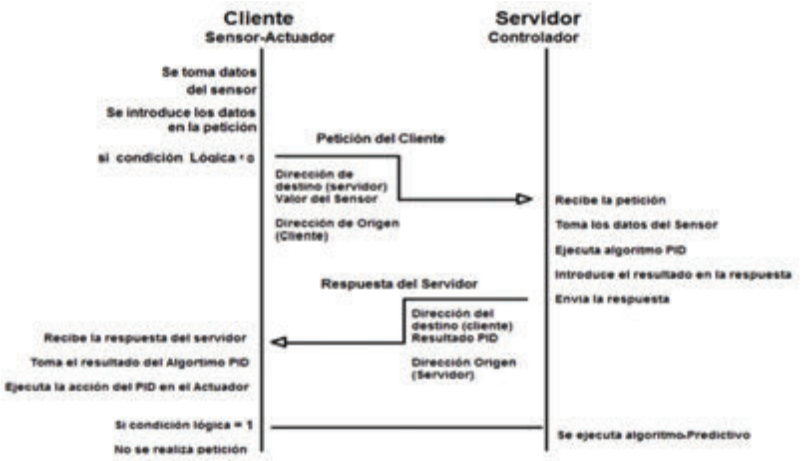
Figura 34
Dragnino Yun como cliente

```
void loop() {

HttpClient client;
sensor();
envio = "http://192.168.0.155/dist:";
envio += distancia;

if (seguro== 0){

client.get(envio);
```

Fuente: los autores

Existe también un seguro que permite realizar peticiones al servidor solo si ciertas condiciones de la planta se cumplen (control por eventos). El cliente es el nodo que inicia la comunicación, el tiempo que existe entre actualización de información es de tipo asíncrono por la naturaleza de la red, incluso si no se llegan a cumplir las condiciones que activan el evento este tiempo puede ser infinito:

Figura 35
Comunicación cliente-servidor



Fuente: los autores

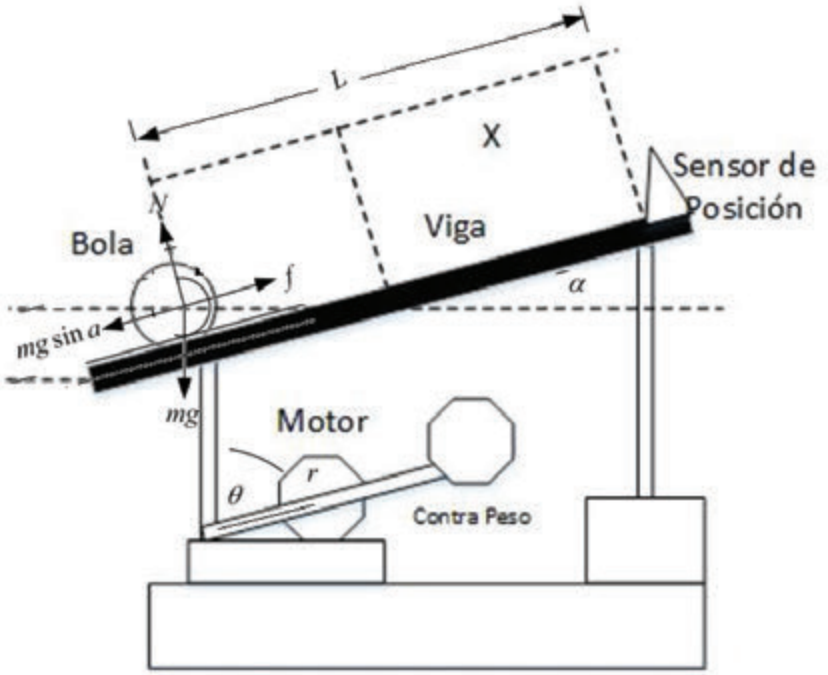
El modelo del sistema Beam-Ball

El sistema Beam-Ball (BB) es un sistema dinámico clásico el cual se ha utilizado para indicar teorías de control y aplicar técnicas, la importancia del sistema BB radica en que es un sistema simple, no lineal e inestable en lazo abierto. Consiste en equilibrar la bola en una barra en una posición deseada, para estabilizar la bola se necesita un sistema de control realimentado, para lo cual se debe medir la posición de la bola y ajustar el motor que sujeta la barra a un ángulo requerido, que vaya acorde al diseño deseado.

Para resolver esta situación muchas estrategias de control se han utilizado: controladores PID (Sridharan, 2002; Ortiz, 2005), control de realimentación de estado, control difuso, control en redes neuronales, etc. pero todas las técnicas anteriores se han considerado en un entorno de trabajo local. En este trabajo se utiliza la técnica de control PID basado en eventos, la cual es una técnica que se basa en un entorno de trabajo distribuido, en donde la planta y el sensor-actuador están en una posición y el controlador está en otra distinta, alejada, pero comunicándose vía red inalámbrica.

El sistema BB considerado en este trabajo está representado en la figura 36. Como puede observarse, se compone de una viga que es sostenida por dos brazos, el uno fijo y el otro móvil, la cual se puede inclinar accionada por un servomotor DC; encima está una bola que puede rodar hacia adelante o hacia atrás dependiendo del ángulo de inclinación. El objetivo del control del sistema BB es encontrar el ángulo adecuado para que la bola pueda permanecer a una posición deseada x, ya que cuando se cambia el ángulo, la acción de la gravedad hará que la bola ruede a lo largo de la viga. El sensor de posición es el que proporciona la distancia de la bola a la referencia del brazo fijo.

Figura 36
Esquema del sistema BB



Fuente: los autores

El modelo simplificado de espacio de estados, de acuerdo con las leyes de la física y aplicando matemáticas, se deriva para el sistema BB aplicando equilibrio de fuerzas mediante la ley de Newton:

$$m\ddot{x}(t) = mgsin\alpha(t) - \mu mgcos\alpha(t) \quad (3.1)$$

Donde m es la masa de la bola, $x(t)$ es la posición de la bola en la viga, g es la aceleración de la gravedad, $\alpha(t)$ es el ángulo de la viga respecto a la horizontal y μ es la constante de fricción cuando la bola rueda por la viga.

Debido a que $\alpha(t)$ y μ son relativamente muy pequeños, casi siempre se asume que: sin $\alpha(t) \approx \alpha(t)$, $\mu \approx 0$, por lo tanto, de la ecuación anterior se tiene que:

$$\ddot{x}(t) = g\alpha(t) \tag{3.2}$$

El ángulo del brazo móvil $\theta(t)$, acorde a la relación geométrica dada por:

$$\alpha(t) = \frac{r}{L}\theta(t) \tag{3.3}$$

Donde r es el radio de palanca al brazo móvil y L es la longitud de la viga. Al sustituir (3.3) en (3.2) se da:

$$\ddot{x}(t) = g\frac{r}{L}\theta(t) \tag{3.4}$$

Suponiendo que el vector que describe el estado es: $x(t) = [x(t)\, \dot{x}(t)\, \theta(t)\, \dot{\theta}(t)]^T$ y $u(t) = \ddot{\theta}(t)$. El modelo del sistema BB, considerando un sistema continuo en espacio de estados, se tiene:

$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \\ \dot{\theta}(t) \\ \ddot{\theta}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & gr/L & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t) \tag{3.5}$$

Para el sistema BB considerado en este trabajo r = 0,07 m, y L = 0,456 m, por lo tanto, el sistema de espacio de estados sería calculado con las siguientes ecuaciones de estado:

$$\begin{cases} x(t) = Ax(t) + bu(t) \\ y(t) = Cx(t) \end{cases} \tag{3.6}$$

Donde:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad y = \begin{bmatrix} x(t) \\ \theta(t) \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.7}$$

El modelo del sistema continuo puede ser expresado en forma discreta como:

$$\begin{cases} x(k+1) = A_d x(k) + b_d u(k) \\ y(k+1) = C_d x(k+1) \end{cases} \tag{3.8}$$

Donde A_d, b_d y C_d pueden ser calculados acorde al periodo específico de muestreo de 0,01 segundos.

Metodología para la simulación e implementación

En este apartado vamos a hablar del modelamiento e identificación experimental, lo cual se construye e instrumenta con elementos como sensores estándar, sensores infrarrojos encargados de medir la posición de la bola, el actuador servomotor encargado de posicionar la viga y la unidad de control. En lo que concierne al diseño mecánico, es importante que el pivote pueda equilibrar a la viga, de lo contrario la gravedad genera una no linealidad que hace el control más complicado.

Primero se obtiene la función de transferencia de la planta completa, luego se hace una simulación utilizando el software Matlab, posteriormente los controladores se implementan en el prototipo y finalmente se mide la respuesta de los controladores al sistema ante una entrada de tipo escalón. Metodológicamente, proponemos un modelamiento e identificación del prototipo experimental asumiendo dos subsistemas: motor y bola (Lizarazo y Borja, 2015):

- El primer subsistema está compuesto por un servomotor, la transmisión, el sistema de brazos (uno fijo y otro móvil) y la viga.

Se considera la entrada como el ciclo útil del PWM del motor $l(s)$ y la salida $\theta(s)$, el ángulo de la viga, en radianes, con respecto a la horizontal. Se va a referir a este sistema como "sistema del motor".

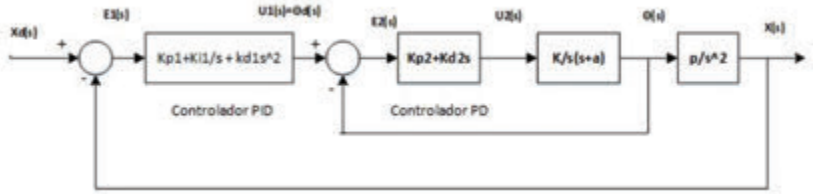
- El segundo subsistema está compuesto por la bola. Se considera como la entrada el ángulo de la viga con respecto a la horizontal $\theta(s)$ y la salida la posición de la bola en la viga $X(s)$. Se va a referir a este sistema como "sistema de la bola".

Debido a que el peso de la bola no es muy grande, la posición de esta no afecta al sistema del motor, por lo tanto, se modelan los dos sistemas independientemente y la salida del sistema del motor es, sencillamente, la entrada del sistema de la bola. El modelo matemático del sistema BB está dado por la combinación del modelo del motor con carga y el modelo de la bola, como se indica en Santos (2014), es decir:

$$\frac{\theta(s)}{l(s)} = \frac{k}{s(s+a)} \quad ; \quad \frac{X(s)}{\theta(s)} = \frac{\rho}{s^2} \tag{3.12}$$

De acuerdo a la ecuación anterior, el modelo de la planta tiene tres polos, por lo que el sistema es inestable en lazo abierto. Por ello, el objetivo de diseñar un controlador es precisamente conseguir la estabilidad en lazo cerrado. En la figura 37 se muestra el diseño del controlador en diagrama de bloques para el sistema BB.

Figura 37
Diagrama de bloques para el control de BB



Fuente: los autores

Para el sistema BB se utiliza dos controladores: el controlador de tipo PID (controlador lazo externo) es el que permite posicionar la bola en el punto deseado y el controlador de tipo PD (controlador lazo interno) permite posicionar horizontalmente el riel. La nomenclatura que se utiliza el para el sistema BB es la siguiente:

- $X_d(s)$ = posición deseada de la bola.
- $X(s)$ = posición de la bola.
- $\theta_d(s)$ = posición deseada del riel.
- $\theta(s)$ = posición del riel.
- $E_1(s)$ = error de posición de la bola.
- $E_2(s)$ = error de la posición del riel.
- $U_1(s)$ = señal de control del controlador PID.
- $U_2(s)$ = señal de control del controlador PD.
- $kp1$, $kp2$ = ganancias proporcionales.
- k_{i1} = ganancia del integrador.
- $kd1$, $kd2$ = ganancias derivativas.

Para el sistema BB se desea que $\lim_{t\to\infty} x(t) = x_d$ donde x_d es una constante, se desea que la función de transferencia de lazo cerrado $X(s)/X_d(s)$ sea constante para que el sistema sea estable. Por lo tanto, el problema es encontrar las ganancias del controlador primario, así como del secundario, para que estabilicen el sistema.

Para el modelamiento se identificó las respuestas del sistema en conjunto, para lo cual se midió la señal de salida del sistema, es decir, la distancia de la bola al eje respecto a una señal de referencia. Estos datos fueron utilizados para la identificación del sistema utilizando el *toolbox* de Matlab (IDENT).[1] De los modelos identificados, se seleccionó la función de transferencia, que representa mejor el comportamiento del sistema, así como la función del PID tomada de los datos experimentales con las contantes $K_p = 8$, $K_i = 0{,}5$ y $K_d = 2{,}8$. Así, se sabe que la función del controlador es:

1 Para mayores detalles ver el anexo A.

$$PID = \left(K_p + sK_d + \frac{k_i}{s}\right) \tag{3.13}$$

Por lo tanto:

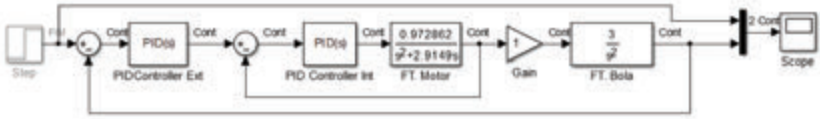
$$Gt = PID{*}FTp \tag{3.14}$$

$$Gt = \frac{6.049}{s^2 + 1.091s + 6.552} \quad ; \quad PID = \frac{2.8s^2 + 8s + 0.5}{s} \tag{3.15}$$

$$FTp = \frac{6.049s}{((s^2 + 1.091s + 0.503) * (2.8s^2 + 8s + 0.5))} \tag{3.16}$$

En este sentido, de forma experimental se obtuvieron las constantes de la función del motor, así como de la bola.

Figura 38
Diagrama de control en cascada simulado

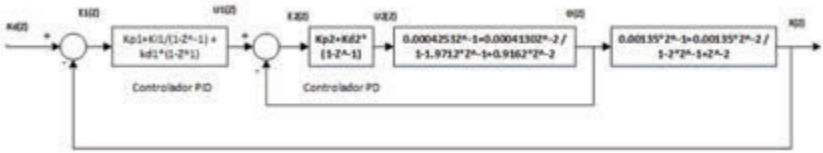


Fuente: los autores

La tendencia actual de controlar los sistemas dinámicos en forma digital en lugar de analógica se debe principalmente a la disponibilidad de dispositivos electrónico-digitales de bajo costo. En la figura 38 se muestra el diagrama de bloques del sistema BB en tiempo continuo, en la transformada de Laplace. En la figura 39 se presenta el sistema en su representación discretizada.[2] Tomando en cuenta que el tiempo de muestreo es de 30 ms, y con los controladores antes presentados, se tiene:

2 En los anexos se muestra cómo se hace dicha transformación.

Figura 39
Sistema BB en su representación Z



Fuente: los autores

Diseño del controlador PID predictivo para la planta BB

Para establecer criterios de predicción en cualquier sistema se debe tener muy claro cómo interactúan las diferentes variables en la respuesta final. Cada variable tiene diferente grado de influencia en el resultado final y dependerá del diseñador considerar las de mayor o menor interacción en el fenómeno de estudio. En la realidad no existen fenómenos lineales, simplemente son aproximaciones. Las funciones de transferencia que expresan el comportamiento de un sistema presentan una gran complejidad, por ende, muchos diseñadores descartan este método para ahorrar varios recursos.

La expresión matemática de un fenómeno es de vital importancia para elaborar estrategias, soluciones y algoritmos de predicción para sistemas de control. En su esencia, la estrategia de control predictivo hace uso de un modelo matemático interno y de una estrategia de optimización para predecir las salidas del sistema dentro de un intervalo de tiempo al que se le denomina "horizonte de predicción".

El movimiento de la pelota sobre el riel es influenciado por varios factores, los más importantes son el ángulo del servomotor al que está sujeta la riel y el peso de la pelota. Este es el criterio central para poder elaborar un algoritmo que nos ayude a calcular un resultado futuro. El controlador tiene acción directa sobre el ángulo del servo, es decir, sobre la causa, permitiendo intuir el valor del efecto (la posición de la pelota sobre el riel).

Algoritmo predictivo para el control PID basado en eventos para el sistema BB

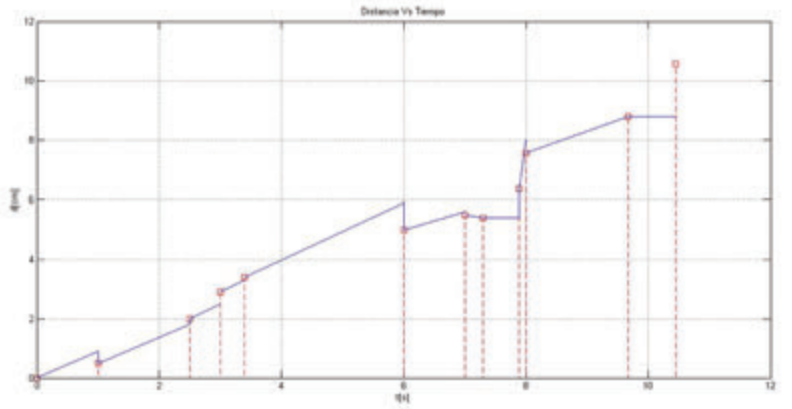
Se desea implementar un algoritmo predictivo para un sistema BB, el cual tiene los sensores y actuadores físicamente alejados del controlador PID basado en eventos, lo que implica que la acción de control se ejecutará cuando ocurra un evento. En este caso el disparo del evento sucederá en el sensor y se dará en base al error entre las lecturas de la señal de salida del sensor (distancia) anterior respecto al actual; en base a este dato y si cumple una función lógica, el sensor seguirá emitiendo datos que ingresarán al algoritmo localizado en el controlador, permitiendo de esta manera calcular la función PID, la que emitirá un dato que permita controlar la posición de la bola a una distancia predefina en la viga. El controlador calcula el valor del ángulo del servo resultado del algoritmo PID. Durante la ausencia de información entre planta y controlador (por ser un sistema de control asíncrono) se pierde información de vital importancia para que algoritmo de control se ejecute correctamente, obteniéndose un sistema oscilante (*cf.* "Alternativas para el controlador PID basado en eventos", cap. 2).

Definición de los requerimientos

Para realizar el algoritmo PID se requiere un dato emitido por el sensor de distancia inalámbrico. Con este dato, al ingresar al algoritmo se realizará el cálculo y minimización del error para efectos de estabilizar el sistema y se emitirá un dato que es el que se enviará al actuador vía Wi-Fi. Durante la ausencia de datos se ingresan valores al PID que no vienen directamente de la planta, sino que son calculados por el controlador basándose en los últimos datos que llegaron:

Los últimos valores de distancia —que la planta envió— y ángulo —que calcula el PID en el controlador— son usados como parámetros para determinar una función lineal que se dispara cuando no existe comunicación entre cliente-servidor.

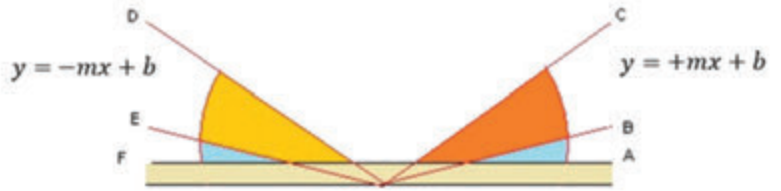
Figura 40
Muestreo asíncrono con información adicional durante la ausencia de comunicación



Fuente: los autores

Para predecir la distancia se usa un contador que inicia y vuelve a cero entre actualizaciones de datos. El último valor de distancia (*disf*) es usado como un *offset* para que la función lineal inicie desde el último valor real. Para definir la pendiente de la función lineal se utiliza el ángulo de inclinación del riel, definiendo la dirección y la velocidad con que la pelota se moverá.

Figura 41
Dirección de la pelota por inclinación del riel



Fuente: los autores

En la figura 41 se presentan los posibles ángulos de giro del servomotor que están comprendidos en los siguientes valores: [F, D] U [C, A]. Cuando el valor del ángulo esté comprendido en el intervalo [C, B], la dirección de la pelota será para la izquierda; cuando el intervalo sea de [E, D], la dirección de la pelota será para la derecha. Para los valores de los ángulos que están en el intervalo [F, E] U [B, A] la pelota no tendrá un movimiento significativo. Con estos valores podemos determinar el signo de la pendiente: [C, B] pendiente positiva, [E, D] pendiente negativa, [F, E] U [B, A] pendiente con valor 0.

Identificación de los módulos del algoritmo implementado

Los módulos implementados son:

- Configuración de la comunicación.
- Definición de variables.
- Inicio de algoritmo.
- Configuración de envío-recepción de datos.
- Lectura y delimitación del dato requerido.
- Conversión de variable.
- Cálculo del PID.
- Registro e impresión de datos para análisis.
- Fin

El algoritmo deberá cumplir con las siguientes características:

- Finitud. El algoritmo termina tras un número finito de pasos.
- Definibilidad. Se define paso a paso de forma clara.
- Entrada. Se definen las entradas necesarias y las variables.
- Salida. Se define e identifica claramente la salida.
- Efectividad. Se define su efectividad por los registros de salida, los cuales se los puede graficar.

Tabla 3
Algoritmo nodo servidor-controlador

1	Configuración de comunicación: servidor en Arduino Ethernet y cliente Wi-Fi en Dragnio Yun.
2	Definición de protocolo de comunicación HTML.
3	Entrada: $X_t^1 = \sum x_i^1$. Lee y almacena variable de distancia de sensor.
4	Calcula el PID en base a los datos de lectura del sensor llegados luego de cumplir una función lógica $Z_{t'}^1 = Z_\tau^1(\hat{x})$; . Procesa la información en el algoritmo de control PID con los datos estimados luego de realizar una periodicidad entre eventos: datos reales llegados.
5	Salida: $Z_{t+\tau}^1$. Envió del valor PID ángulo al actuador remoto, dato enviado vía Wi-Fi.

Fuente: los autores

Algoritmos implementados en el WNCS

El principal problema de un sistema de control en red tipo WNCS es su tiempo de muestreo variable. Para el lado de la planta, en la cual está el sensor-actuador que actúa como cliente en la red, el periodo puede ser fijo; pero para el lado del controlador, que actúa como servidor que puede ser en una LAN o una WAN, el periodo es de naturaleza no determinística debido a que la red de comunicación actualiza la información de una forma aleatoria. En los procesos de control el periodo de muestreo es fundamental para tener un óptimo desempeño del algoritmo del controlador. Cuanto más sensible es la variable medida ante las acciones del actuador en el tiempo, se requiere de un menor tiempo de muestreo. Por esta razón se desarrollaron las redes industriales, que son de carácter determinísticos. Existen dos áreas que presentan posibles soluciones: a) comunicaciones: control por eventos, incremento de ancho de banda, subredes dedicadas, etc. b) control: algoritmos de control acoplados para trabajar sin tiempo fijo de actualización de datos, algoritmos predictivos, etc.

Algoritmo en el servidor-controlador predictivo PID asíncrono

En el algoritmo PID el tiempo de muestreo (*T*) permite calcular el componente integral y derivativo (figura 42). Se calcula el componente integral con la fórmula del área del trapecio, acumulando cada valor del área en la misma variable. *T* es la altura del trapecio, *error* es la base mayor y *error_anterior* sería la base menor. El factor derivativo *T* es la variación de tiempo que divide la variación de error.

Figura 42
PID asíncrono 1

```
void loop() {

 error = sp - distancia;
  proporcional=error;
  integral = integral + ((error+error_anterior)/2.0)*(T);
  derivativo = (error - error_anterior)/(T);

delay(T)

}
```

Fuente: los autores

La variable *T* es fija en su sistema clásico de control, teniendo al error como el valor que cambia entre cada actualización de datos. En un sistema de control en red, el cliente (planta) realiza peticiones al servidor (controlador), el cual responderá con la respuesta del PID. El tiempo entre peticiones y respuestas es de tipo variable, es decir, *T* cambia con cada actualización de datos. Para el cálculo PID se tienen dos valores que cambian constantemente entre cada actualización de datos.

En un PID asíncrono es necesario obtener el valor de *T* para el cálculo del PID. Para tener el valor de *T* se usa un contador (*dt*) que aumenta cada milisegundo y en el que se mantiene el tiempo de ac-

tualización de información. El servidor, al recibir información, usa este valor almacenado para el cálculo PID y luego lo vuelve a cero para una futura petición:

Figura 43
PID asíncrono 2

```
void loop() {

 EthernetClient client = server.available();
  if (client) {

  Servidor();
  error = sp - distancia;
  proporcional=error;
  integral = integral + ((error+error_anterior)/2.0)*(dt/1000);
  derivativo = (error - error_anterior)/(dt/1000);
  pid=(kp*proporcional)+(ki*integral) +(kd*derivativo);
  error_anterior=error;
   dt=0
  }


delay(1)
dt++
}
```

Fuente: los autores

Algoritmo predictivo con PID síncrono y semi-síncrono

En un sistema donde la razón de cambio de la variable medida respecto al tiempo es de un reducido valor, el PID asíncrono funciona óptimamente aun con la pérdida de información. La acción del actuador no influye significativamente en el sistema en un corto tiempo. Sin embargo, existen sistemas que cambian sus parámetros en cortos lapsos

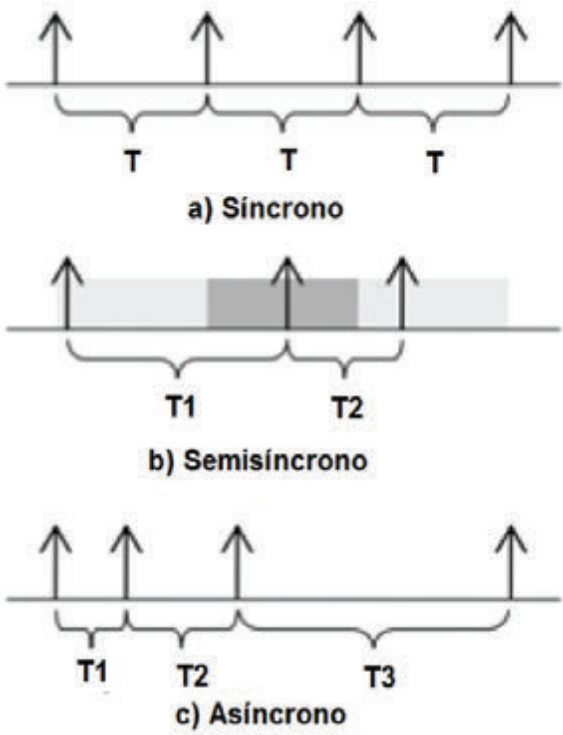
de tiempo. Este es el principal problema de un sistema donde la red es no determinística, la pérdida de información entorpece la acción de control. Toda la teoría de control justifica su trabajo en sistemas continuos o con periodos de muestreo fijos, aquí es donde se incorpora una nueva forma de acoplar un sistema totalmente asíncrono a uno síncrono o semi-síncrono, incorporando información en base a las respuestas del controlador durante el tiempo de actualización.

Durante la ausencia de datos el PID asíncrono necesita de información extra (tendencia) que se aproximarán a la realidad incorporando datos en función de los resultados reales. Se introducirán valores ficticios para que el algoritmo de control trabaje incluso durante la ausencia de datos. Sin estos datos adicionales la respuesta del PID es abrupta y torpe, sin equilibrar el sistema. BB es un sistema que requiere de tiempos de muestro muy bajos, alrededor de 100 ms, pues la variable medida cambia drásticamente en cortos lapsos de tiempo, razón por la cual se decidió trabajar con este sistema.

El muestreo semi-síncrono tiene un periodo fijo que esporádicamente aparecerá, dependiendo de la naturaleza del sistema. Este periodo fijo no es constante, pero al realizar un promedio de los tiempos de muestreo se observa que se encuentra próximo a este valor:

Para convertir un sistema asíncrono en semi-síncrono o síncrono es necesario trabajar con información adicional durante la ausencia de datos reales. La función lineal estará constantemente trabajando en el sistema sin importar si la planta envía o no información. El PID tomará siempre los datos entregados por la función lineal, que están basados en los datos de salida y entrada del sistema (distancia y ángulo). De cierta manera, el sistema no trabaja con datos auténticos, sino con una tendencia que presenta un error mínimo respecto a la realidad.

Figura 44
Tipos de muestreo



Fuente: los autores

Algoritmos en el cliente sensor-actuador

Un evento se puede definir como una condición lógica que bajo ciertas condiciones entrega un 0 o 1 lógico al sistema. Dependerá del diseñador establecer los criterios y parámetros de trabajo de la condición, ya que no existe una norma definida y el método de calibración es basarse directamente en pruebas y errores al trabajar con el sistema. Esta condición directamente controla el envío-recepción de datos, liberando el medio de comunicación de trabajo innecesario.

Al trabajar bajo la arquitectura cliente-servidor, el cliente es el que tiene el control del medio de comunicación. Al realizar una petición, el servidor responderá inmediatamente, por el contrario, bajo la ausencia de la misma, se detendrán las comunicaciones. La condición permite que se realicen o no peticiones al servidor.

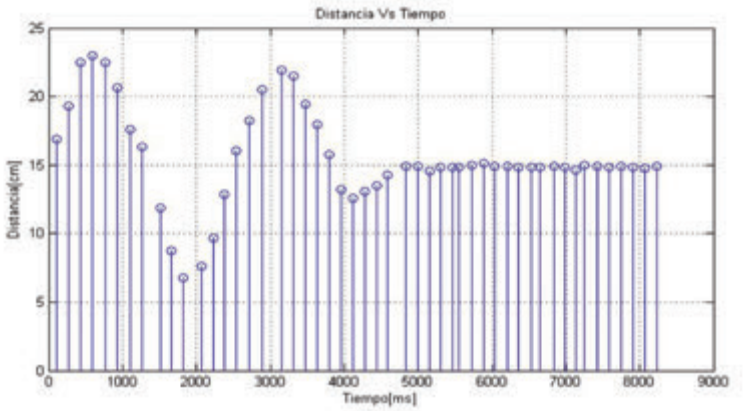
A continuación se analizarán las diferentes condiciones aplicadas. Cabe destacar que para el estudio de eventos se usó el PID asíncrono con el objetivo de observar la funcionalidad del sistema con los retardos propios de la red. Los diferentes límites y variaciones para las condiciones de eventos fueron tomados observando que cumplan con los requisitos de equilibrio del sistema. El PID trabajó con los siguientes coeficientes: $K_p = 2{,}65$, $K_i = 0{,}01$ y $K_d = 0{,}75$.

Condición intervalo de trabajo temporizado

Se establece una franja de trabajo para activar la condición lógica cuando la variable a controlar (distancia de la bola al punto de equilibrio) se encuentre dentro de un rango cercano al *set point*. El rango seleccionado dependerá de la precisión que se desee lograr, pues influirá en la ocupación de la red. Un contador se activa cuando se encuentra dentro del rango, alrededor del punto de equilibrio, cuando alcanza un límite de tiempo definido dejará de realizar peticiones. En el cliente se programa la condición de disparo que permite enviar peticiones al servidor con los datos de la planta.

La condición de disparo y el contador regresan a su estado inicial cuando una perturbación le haga salir de la franja de trabajo, permitiendo que se realicen nuevas peticiones para estabilizar el sistema. En la figura 45 se puede observar el funcionamiento del PID: el periodo de actualización varía dependiendo de la red, el sistema se estabilizó a los 5 250 ms y el pico máximo es de 22,98 cm. Se observa que el contador detiene automáticamente el envío de datos y que el proceso duró alrededor de 8 229 ms.

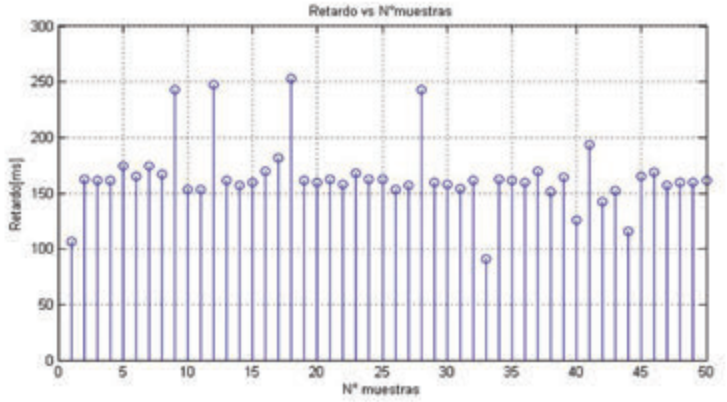
Figura 45
PID asíncrono con intervalo de trabajo temporizado



Fuente: los autores

En la figura 46 se observan los retardos durante cada actualización de datos.

Figura 46
Retardo del PID asíncrono con intervalo de trabajo temporizado



Fuente: los autores

Tabla 4
Datos de los tiempos de retardo en el PID asíncrono con intervalo de trabajo temporizado

Máximo tiempo de retardo (ms)	253
Mínimo tiempo de retardo (ms)	91
Promedio (ms)	164,58

Fuente: los autores

Condición de variación de la variable medida

El controlador PID tiende a equilibrar el sistema. Se observa que las oscilaciones del sistema tienen una menor magnitud en el tiempo. La razón de cambio entre el valor de la variable medida actualmente $y(t)$ y la anterior $y(t_{last})$ tiene un valor menor al acercarse al *set point*.

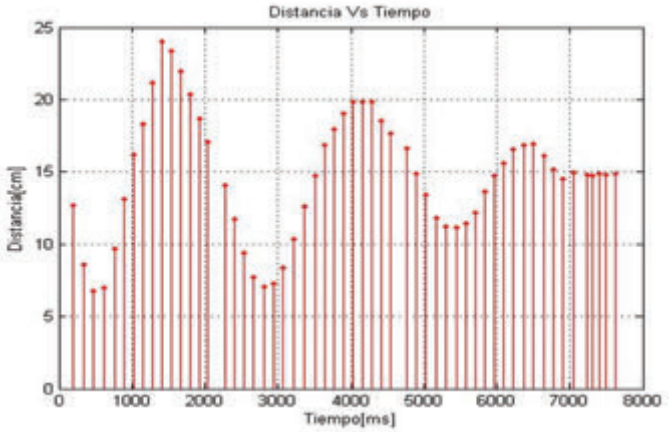
$$|y(t)\text{-}y(t_{last})| \geq \Delta \tag{3.17}$$

Es lógico pensar que una vez equilibrado el sistema la razón de cambio o variación será cero. Esta condición es usada para disparar el evento que activa o desactiva las comunicaciones entre cliente y servidor.

A diferencia de la condición anterior no existe tiempo, pero sí hay una franja de trabajo. Esta franja donde se activa la condición de variación se encuentra situada entre intervalos del *set point*. Se programa la variación entre cada actualización de datos en la variable de distancia *vd*, la diferencia entre el valor actual y el anterior se almacenan en esta variable. La franja se continúa usando debido a que pueden existir variaciones muy pequeñas en distancias muy alejadas al *set point*, deteniendo al sistema innecesariamente. Al no cumplir alguna de las condiciones, el sistema solicitará nuevamente una petición al controlador.

En la figura 47 se puede observar el funcionamiento del PID. Aquí el periodo de actualización varía dependiendo de la red. El sistema se estabilizó a los 6 903 ms, el pico máximo es de 23,98 cm y el tiempo de trabajo total es 7 630 ms.

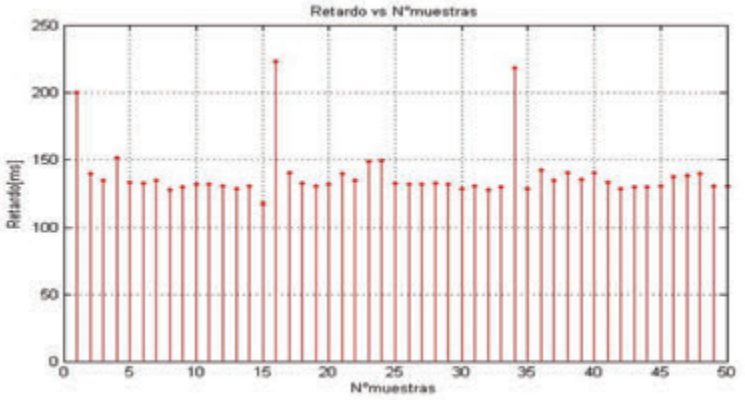
Figura 47
PID asíncrono con variación de variable medida



Fuente: los autores

En la figura 48 se observan los retardos durante cada actualización de datos.

Figura 48
Retardos del PID asíncrono con variación de la variable medida



Fuente: los autores

Tabla 5
Datos de los tiempos de retardo en PID asíncrono con variación de la variable medida

Máximo tiempo de retardo (ms)	223
Mínimo tiempo de retardo (ms)	117
Promedio (ms)	138,22

Fuente: los autores

Condición de acumulación del error

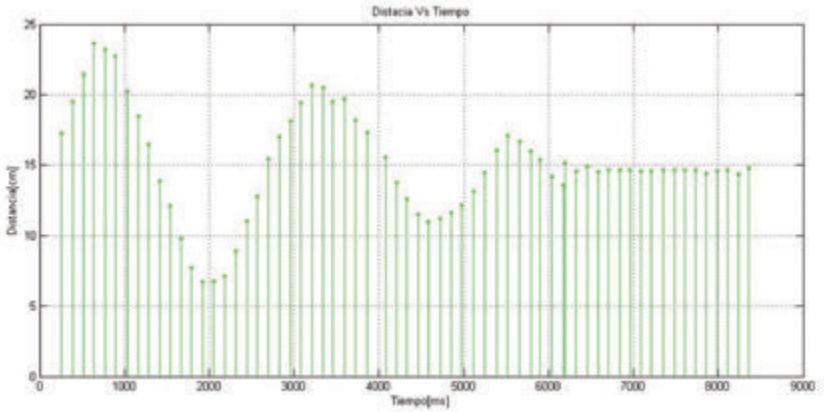
El error es el parámetro que activará o desactivará el evento. Es lógico pensar que cuando el sistema tiende a equilibrarse, el valor del error disminuye paulatinamente. Al tener un sistema oscilante, el error aumentará y disminuirá constantemente, es por esta situación que la acumulación del error solo entra a trabajar en una franja de trabajo.

$$\sum e(t) > \Delta \tag{3.18}$$

El programa es exactamente el mismo, con la diferencia de que la condición que activa o desactiva el evento trabaja con la acumulación del error. En la figura 49 se puede observar el funcionamiento del PID: el periodo de actualización varía dependiendo de la red, el sistema se estabilizó a los 6 195 ms, el pico máximo es de 23,94 cm y el tiempo de trabajo total es 8 365 ms.

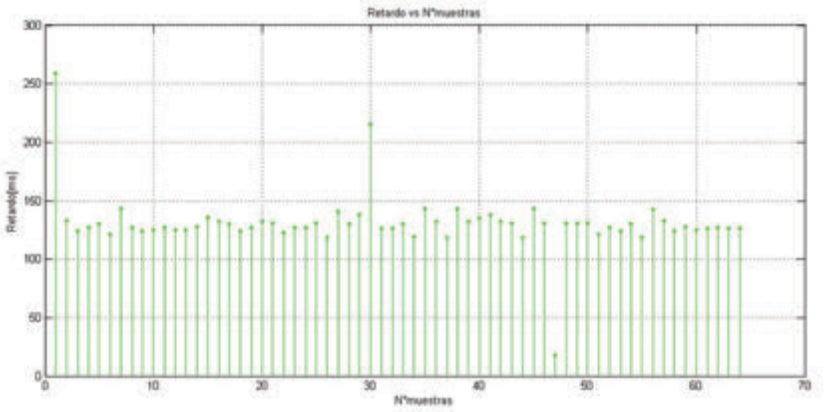
En la figura 50 se observan los retardos durante cada actualización de datos.

Figura 49
PID asíncrono con error integral



Fuente: los autores

Figura 50
Retardos del PID asíncrono con error integral



Fuente: los autores

Tabla 6
Datos del tiempo de retardo en PID asíncrono
con acumulación del error

Máximo tiempo de retardo (ms)	259
Mínimo tiempo de retardo (ms)	18
Promedio (ms)	130,7

Fuente: los autores

Condición de acumulación del error al cuadrado

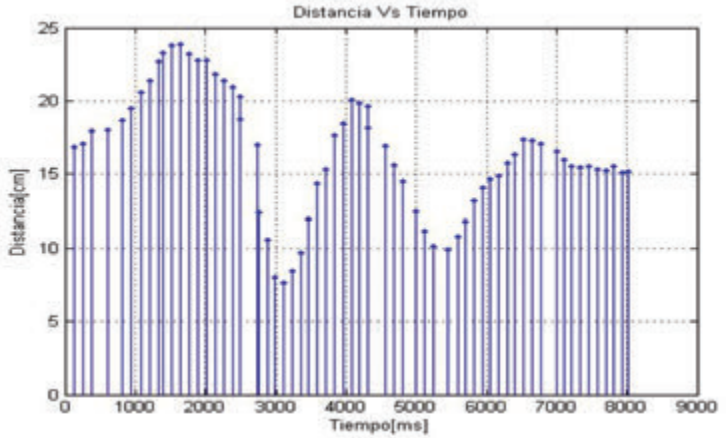
El error es el parámetro que activará el evento. A diferencia de la condición anterior, el error es elevado al cuadrado. Cuando la variación de este parámetro exceda un valor determinado, la planta detendrá las peticiones al servidor.

$$\sum e^2(t) > \Delta \tag{3.19}$$

El valor del error integral al cuadrado para activar el evento aumenta a 11 para estabilizar el sistema. En la figura 51 se puede observar el funcionamiento del PID: el periodo de actualización varía dependiendo de la red, el sistema se estabilizó a los 7 217 ms, el pico máximo es de 23,89 cm y el tiempo de trabajo total es 8 020 ms. a partir de esto, el tiempo trabajó innecesariamente sería: *TI* = tiempo de trabajo – tiempo de estabilidad, entonces, *TI* = 803 ms.

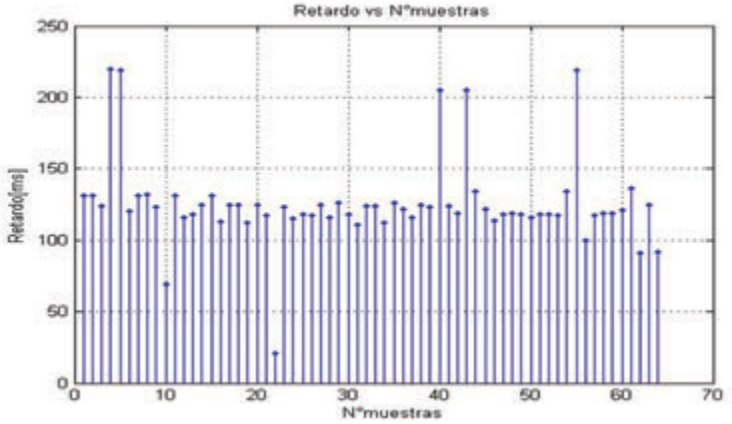
En la figura 52 se observan los retardos durante cada actualización de datos.

Figura 51
PID Asíncrono con error integral al cuadrado



Fuente: los autores

Figura 52
Datos de los tiempos de retardo en PID asíncrono
con error integral al cuadrado



Fuente: los autores

Tabla 7
Datos del tiempo de retardo en PID asíncrono
con error integral al cuadrado

Máximo tiempo de retardo (ms)	227
Mínimo tiempo de retardo (ms)	21
Promedio (ms)	125,31

Fuente: los autores

Definición de la mejor condición para el evento del sistema

Al trabajar con un sistema de control en una red no determinística, el tiempo de estabilidad y trabajo son los parámetros que nos permiten definir qué condición usa mejor los recursos del sistema, es decir, el algoritmo tiene la inteligencia necesaria para pedir o no información a la central de control.

Una de las grandes ventajas de control por eventos es ahorrar recursos del sistema. El evento a más de detener la comunicación entre planta y controlador, también detiene los cálculos o algoritmos que no son necesarios cuando el sistema ha alcanzado el equilibrio. El tiempo de estabilidad es variante, pero esto no depende de las condiciones de los eventos, esto se debe a la naturaleza aleatoria del tiempo de muestreo. El tiempo innecesario de trabajo varía dependiendo de la condición para el evento. Se puede observar en la tabla 8 que el menor valor es el de la condición de variación de la variable medida y por obvias razones es la que más recursos ahorra al sistema.

Tabla 8
Datos de las condiciones para eventos

Condición	Tiempo de trabajo (ms)	Tiempo de estabilidad (ms)	Retardo promedio (ms)	Máximo retardo (ms)	Tiempo innecesario de trabajo (ms)
Intervalo de trabajo temporizado	8 229	5 250	164,58	253	2 979
Variación de la variable medida	7 530	5 003	138,22	223	727
Acumulación del error	8 365	5 195	130,7	259	2 170
Acumulación del error al cuadrado	8 020	7 217	125,31	227	803

Fuente: los autores

Capítulo cuatro
Implementación de un WNCS local y remoto para el sistema BB mediante Raspberry Pi y Arduino

En el presente capítulo se describirá el procedimiento para implementar un WNCS mediante servidor Raspberry Pi-3, en donde estará el algoritmo controlador PID y el cliente, es decir, la planta conformada por un sensor de distancia, además de un actuador como el servomotor, los cuales son controlados por un Arduino Mega con una Shield Dragino Yun para la comunicación inalámbrica.

Dentro del Raspberry Pi-3 se tiene un servidor LAMP (Linux, Apache, MYSQL, PHP), donde se programa la adquisición de datos, el cálculo del controlador en base a la función de transferencia de la planta, el envío de los datos obtenidos y calculados a una base de datos en un archivo PHP, la encapsulación de datos a través de un método de escritura ligera o intercambio de información (JSON) y finalmente la visualización del comportamiento a través de HTML y JavaScript que obtienen sus datos desde una base de datos MYSQL.

Al Arduino Mega con Shield Dragino Yun se envían los datos del sensor, los cuales se adquieren y escalan con el objetivo de únicamente dar la información requerida, que en este caso es la distancia en centímetros (cm). Después realiza el proceso de interpretación de los datos calculados que recibe del servidor para dar un ángulo de movimiento al actuador, que es un servomotor.

Descripción de los dispositivos del sistema

Servidor Raspberry Pi-3

Raspberry es un computador compacto de placa reducida de tecnología SMD (Tecnología de Montaje Superficial). Se lo puede considerar como un hardware parcialmente libre, ya que no todos los componentes del Raspberry están bajo contrato, pues está orientado a la educación informática, la cual actualmente es utilizada en muchos servicios de IoT (Internet of Things). El propósito de este dispositivo es más académico y de desarrollo informático y robótico (Raspberry, 2017). Usa un software libre que es una adaptación de Debian, un sistema más ligero para poder trabajar con una tarjeta micro-sd de 8 Gb mínimo, y para nuestro proyecto se le ha llamado Raspbian Jessi 2017.

La ventaja de usar el Raspberry Pi-3 para la implementación de la comunicación inalámbrica como servidor de un proceso de control, es que al usar el sistema operativo Raspbian los permisos para generar un servidor web son más accesibles y no se requiere software ni complementos adicionales que no sean libres.

La comunicación inalámbrica se obtiene de un formato ligero de intercambio de datos denominado JSON, que significa: JavaScript Object Notation (notación de objetos de JavaScript) (ECMA-404, 1999), donde se realiza el intercambio de objetos entre el Raspberry y la planta, que en este caso es implementada en la Shield Dragino Yun.

El cliente

El cliente es un dispositivo conjunto, ya que por sí solos los elementos de este dispositivo no funcionarían. El Shield Dragino Yun es un complemento del Arduino que mejora tanto su procesamiento como su memoria interna, para soportar la interacción inalámbrica entre el servidor y la planta, la cual viene a ser el sensor y el actuador inalámbrico.

- Arduino Mega. Se trata de una tarjeta controladora, basada en los At-megas, que tiene varias aplicaciones. Actualmente es un dispositivo de fácil manejo y programación con grandes prestaciones, compatible con software y hardware libre como son los *shields* que están diseñados para estos dispositivos y que facilitan la comunicación entre dispositivos.
- Dragino Yun. Es un dispositivo desarrollado para resolver los problemas de conexión y almacenamiento que tiene el Arduino. Es compatible con IDE de Arduino ya que tiene un software de código abierto que se ejecuta en el *shield* para configurar el mismo. Este dispositivo es potente ya que existen aplicaciones web en las cuales al mismo dispositivo se lo puede usar como servidor o cliente. Se pueden cargar páginas básicas por su gran capacidad de almacenamiento y versatilidad, además, se puede conectar a la red en la que se trabaja localmente de forma inalámbrica.

Herramientas para implementar algoritmos a través de la nube

PHP (Hypertext Preprocessor)

Es un lenguaje de programación para desarrollo web. Con la programación PHP se ejecutan comandos de almacenamiento en la base de datos mientras se procesa la desencapsulación y el cálculo del PID para guardar la información en tiempo real, tanto de la distancia que obtiene como de la respuesta del cálculo realizado, y con eso podemos usar los datos para graficar el comportamiento del sistema.

> PHP es un lenguaje de procesos desarrollado para el lado del servidor ya que tiene características de simplicidad para los programadores y puede ser desarrollado también en HTML. Se utiliza la programación de PHP para desarrollar la des encapsulación de la información que le llega al servidor para posteriormente enviarla a un archivo de texto (archivo. txt), debido a que la configuración en el Raspberry Pi-3, para la lectura

y escritura de un archivo de texto es fácil, habilitando los correspondientes permisos de lectura y escritura de (archivo.txt) (PHP, 2018).

Base de datos MySQL

La base de datos es una forma estructurada de organizar y guardar la información. Se trabaja con esta estructura para realizar las gráficas del comportamiento del sistema a controlar. La información que se obtiene en la página PHP (información del sensor u otras variables del sistema) es guardada en orden de llegada tipo FIFO, la cual se organiza para graficar de forma adecuada el comportamiento del sistema usando la configuración TimeStamp, que es una marca temporal en formato de fecha y hora (Oracle, 2018).

JSON (notación de objetos de JavaScript)

Esta es una forma de serializar la información de variables ya sean numéricas o booleanas. Está basada en el lenguaje de programación JavaScript, de fácil interpretación para humanos y aún más para las máquinas, por lo que nos sirve para intercambiar información rápidamente, gracias a su sencillez de escritura y lectura. JSON tiene dos estructuras: la estructura de pares y la estructura de lista. En nuestro proyecto se utiliza la estructura de pares, que consiste en nombre y valor.

Diseño de la implementación del WNCS

El usuario busca tener control y monitoreo de un controlador PID en una planta BB, el cual está conectado de forma inalámbrica a través de una red LAN, eliminándose los elementos guiados y el cableado entre equipos (figura 53). De esta forma se consigue una mayor potencia y movilidad de los sensores y actuadores.

Figura 53
Esquema didáctico del proyecto

Fuente: los autores

El WNCS implementado virtualiza el controlador físico, el cual puede ser ejecutado de forma local o remota: de forma local se usa un servidor LAMP (Linux, Apache, MySql, Php) en el dispositivo Raspberry Pi-3, como se puede apreciar en la figura 54; de forma remota se usa un alojamiento virtual, como se puede apreciar en la figura 55, alojamiento que se encuentra en un proveedor de servicios web, que para efectos de esta implementación fue SmarterAPS.NET, con dirección IP dada por el proveedor y dirección DNS.

Figura 54
Esquema de control local en Raspberry Pi-3

Fuente: los autores

Figura 55
Esquema de control en la nube

Fuente: los autores

La información que envía el controlador virtualizado se realiza mediante el método de intercambio de información JSON. En cambio, la decodificación de la información es realizada por el Arduino Mega y la Shield Dragino Yun, que está conectada al sensor y actuador de la planta.

Se realiza el cambio de controlador usando un servidor local, el cual es creado en la Raspberry Pi-3 y emula el comportamiento de un servidor Linux local, servidor LAMP (Linux Apache MySql PHP). Una vez que se ejecuta el algoritmo PID en forma local, se procede a subir el controlador a un espacio en la nube para observar el comportamiento del controlador de forma remota.

Arquitectura de la planta

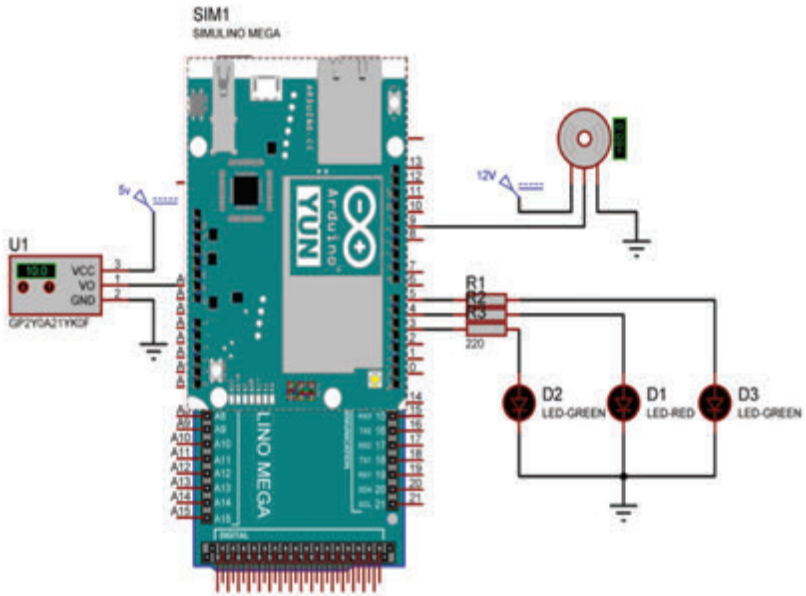
Esquema del cliente

En este caso, la planta contiene al sensor y al actuador que se comunican en forma inalámbrica, en donde se tiene un Arduino Mega el cual adquiere la información de distancia del sensor a través de una entrada analógica que está incorporada en la tarjeta. Esta información

la envía al controlador, el cual calcula el ángulo de inclinación de la barra. Luego de llegar el dato, el cálculo es ejecutado a través de un pin de salida PWM configurado para el servomotor.

El Shield Dragino Yun es un dispositivo que permite hacer la comunicación inalámbrica entre el módem y el Arduino Mega, para poder enviar la información de distancia que es adquirida del sensor. También se encarga de recibir la información del ángulo de inclinación de la barra que es enviada del controlador por la red.

Figura 56
Estructura de Arduino y Dragino Yun

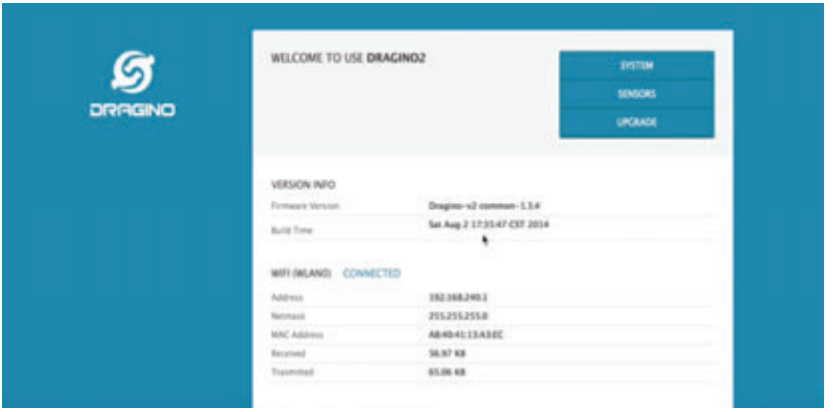


Fuente: los autores

Luego, la configuración general de la Shield Dragino Yun se realiza de la siguiente manera:

- Primero, una vez que se conecta el Arduino Mega con la Shield Dragino Yun, se energiza, y el dispositivo genera una red inalámbrica.
- Se busca la red inalámbrica que genera el dispositivo en el explorador de redes, para conectarnos entre la computadora y el Arduino Mega con Dragino Yun.
- Después de establecer la conexión entre la computadora y el Arduino, se procede a abrir cualquier explorador y en la barra de direcciones se escribe "192.168.240.1", siempre y cuando la computadora esté configurada dentro de la misma red.
- Luego nos aparece una interfaz en la cual se pide la contraseña, que por defecto del equipo es "dragino".
- Entonces se procede a la interfaz principal, donde escogemos la opción "System", como se indica en la figura 57.
- Nos desplazamos hasta la opción "Wireless Parameters", donde buscamos el nombre de la red local a la cual nos vamos a conectar, y en la parte de abajo ponemos el tipo de seguridad que tiene la red y la clave de la misma.
- Ahora, para configurar el software de Dragino Yun e implementar la librería JSON que se descargó, se adiciona la librería al programa Arduino, la cual se llama "ArduinoJson-master.zip". La adición de esta librería se la realiza abriendo la pantalla de ArduinoIDE y seleccionando en la barra superior: "Programa à Incluir librería à Añadir librería Zip".
- Se abre el explorador de archivos y se busca el archivo "ArduinoJson-master.Zip" donde se encuentre guardado. Finalmente, se escoge "Aceptar".

Figura 57
Interfaz principal



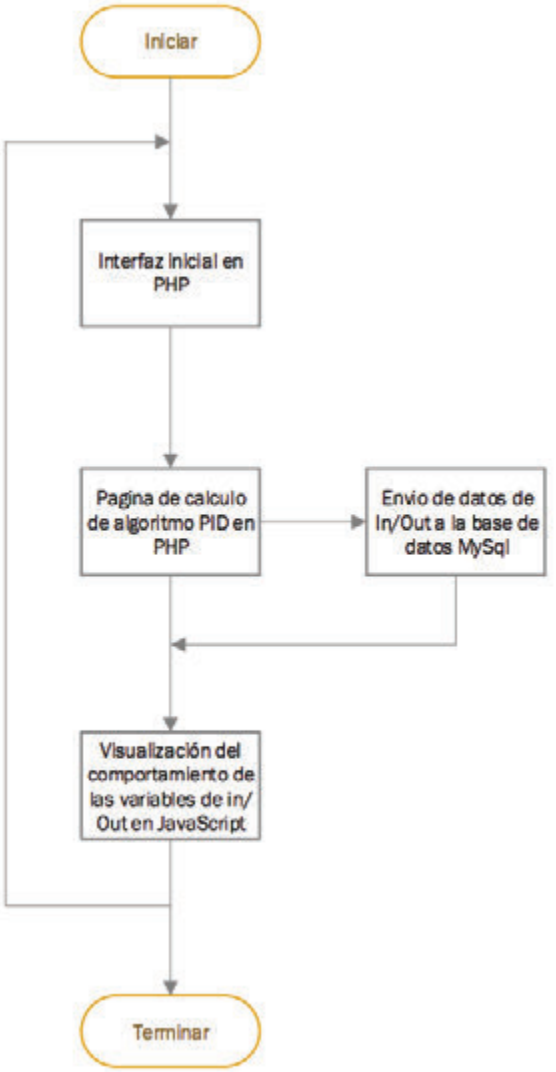
Fuente: los autores

Esquema del controlador

El controlador está implementado sobre un servidor Linux, el cual trabaja con servicios libres y de fácil configuración, ya que no necesita tantos permisos como lo usa el Windows; sin embargo, al ser libre es más vulnerable a ataques del exterior. Para la edición del programa se utiliza un interpretador de código ligero por defecto, que es el Geany Programmer's Editor, el cual soporta varios lenguajes de programación como PHP o JavaScript, con los que se desarrolla el controlador que se utiliza en el proyecto (ver figura 58).

El anterior flujograma describe el funcionamiento global del controlador que se implementó en el servidor. Allí vemos la ejecución del proceso que se realiza en el controlador, el cual adquiere datos de entrada/salida (*in/out*), luego ejecuta el algoritmo PID para la panta BB y finalmente visualiza el comportamiento de las variables a través de una interfaz gráfica realizada en JavaScript.

Figura 58
Flujograma del controlador



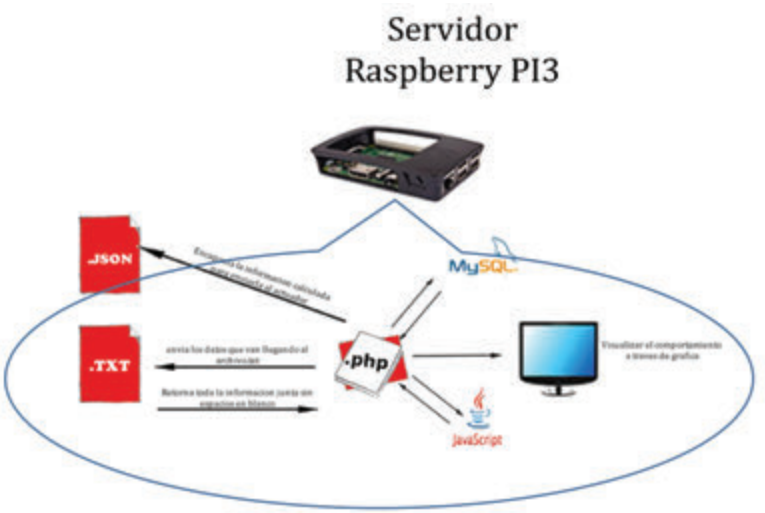
Fuente: los autores

Esquema de funcionamiento local

Para el funcionamiento local se utilizó un Raspberry Pi-3 como servidor, en el que se ejecuta el siguiente procedimiento:

- En la página principal se adquieren los datos obtenidos del sensor y se almacenan temporalmente en un archivo tipotxt.
- Se adquiere el dato almacenado en el archivo temporal anterior y se realiza el cálculo del algoritmo PID.
- Se envía el dato resultante del PID a la base de datos, para posteriormente graficarlo.
- Se realiza el traslado de los datos mediante el formato de intercambio de información JSON.
- Se virtualiza el comportamiento de las variables de entrada del sensor y la variable de salida, que es la respuesta del algoritmo PID, a través de una interfaz gráfica creada en JavaScript.

Figura 59
Esquema del controlador local

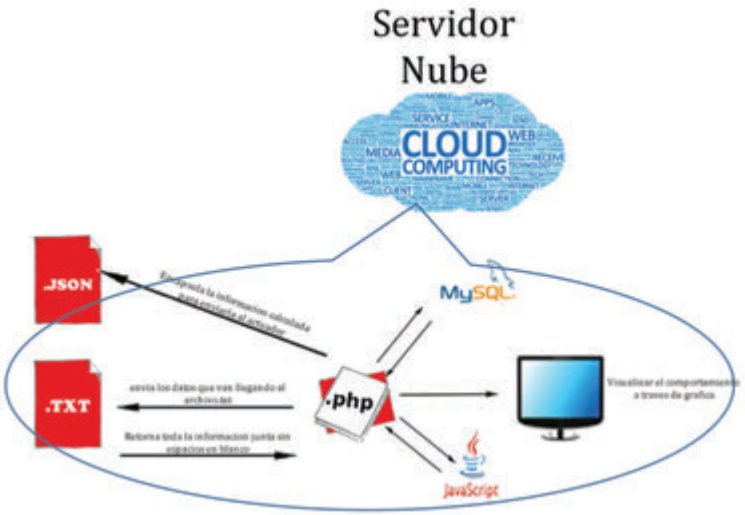


Fuente: los autores

Esquema de funcionamiento en la nube

Para el funcionamiento remoto se utilizó un alojamiento en una nube, en la cual se ejecuta un procedimiento similar al anterior, con la diferencia que al llevar el controlador a la nube se deben trasladar los archivos PHP, JSON y JS. El resto de elementos que se necesitan se los crea en el proveedor de servicios (SmarterASP.NET) ya que al trasladarlos del servidor local al servidor remoto los archivos no se ejecutan.

Figura 60
Esquema de controlador en la nube



Fuente: los autores

Existen restricciones del proveedor de servicios cuando se tiene una cuenta temporal o de prueba que limita la capacidad de la base de datos para almacenar información, lo cual no es de gran impacto en el controlador ya que la base de datos es un respaldo de información para generar reportes gráficos independientes.

Procedimiento para la implementación del sistema

A continuación se describe el procedimiento utilizado para la implementación del sistema. Aquí se expone el esquema del controlador programado en el servidor y la implementación de los algoritmos tanto para el cliente como para el servidor.

Esquema en PHP y MYSQL

La programación en PHP es la más eficiente y utilizada para trabajar con base de datos, ya que se tiene mucha ayuda y librerías de comandos de funcionamiento, así como ejemplos en línea funcionales de este proceso, listos para guardar y obtener datos. Al trabajar con PHP también se trabaja con un lenguaje adaptativo que es muy compatible con diferentes lenguajes de programación como HTML y JavaScript; además de que puede ser el lenguaje web utilizado para nuestra implementación.

- La comunicación comienza con la petición del cliente, el cual envía los datos al servidor donde se realiza el cálculo del algoritmo PID, inicialmente se recibe la respuesta del PID en cero.
- Se establece la comunicación con el servidor entre la interfaz principal y el usuario.
- En la interfaz principal se envían los datos a un archivo temporal como van llegando.
- Del archivo temporal se recibe lo almacenado y se decodifica la información obteniéndose los datos de distancia del sensor, para realizar el cálculo del controlador PID.
- Después de realizar los cálculos del controlador PID se realiza el mapeo de la respuesta, para expresarla en ángulo y enviarla a través del método de intercambio de información JSON.

La aplicación se implementó en PHP, un lenguaje para trabajar con la interfaz de envió de datos y graficar la respuesta del comportamiento del proceso, combinando HTML y JavaScript en la interfaz final.

La figura 61 describe el comportamiento entre los diferentes lenguajes utilizados para la implementación: existe una comunicación bilateral entre MySql y PHP para realizar consultas de los datos almacenados en la base, también existen librerías que permiten realizar reportes gráficos de manera amigable. Cabe recalcar que las librerías son editadas en JavaScript. En nuestro proyecto se utilizó la librería de HighChart, la cual se editó para realizar la visualización de las variables de entrada/salida (*in/out*).

Figura 61
Iteración de lenguajes para el monitoreo

Fuente: los autores

Cuando la interfaz principal entra en funcionamiento, inmediatamente se establece una comunicación PHP y MySql. En MySql se obtienen los datos de la variable de entrada —que es el sensor— y la variable de salida —que es la respuesta del PID—. En la interfaz de presentación se pueden graficar los datos almacenados en la base de MySql para entender el comportamiento del sistema y observar cómo se estabiliza. También sirve usar JavaScript para realizar la interpretación de la información y generar los reportes gráficos.

Procedimiento de configuración del LAMP (Linux, Apache, MySql, PHP)

En la implementación de este sistema se usan varios comandos para crear un servidor LAMP (Linux, Apache, MYSQL, PHP) en Raspberry Pi-3. Todos los comandos son ejecutados en el panel de comandos de Raspbian Jessy 2017. Se debe actualizar la Raspberry antes de instalar los complementos del LAMP.

- Comando de descarga de paquetes de las páginas designadas: sudo apt-get update
- Comando de instalación de los paquetes descargados sin dañar la configuración preestablecida del sistema: sudo apt-get upgrade
- Comando de eliminación de archivos residuales que quedan después de la instalación de las actualizaciones: sudo apt-get autoremove

Se procede a instalar el LAMP Server como se describe a continuación:

- Comando de descarga e instalación de los paquetes Apache de servidor local: sudo apt-get install apache2
- Comando de descarga e instalación de servidor y cliente MySql: sudo apt-get install mysql-server mysql-client
- Comando de descarga e instalación de las librerías de comunicación entre PHP y MySql: sudo apt-get install php5 libapache2-mod-php5 php5-mysql –y
- Comando de reinicio del servicio Apache sin reiniciar el Raspberry Pi-3: sudo service apache2 restart

Se procede a instalar PhpMyAdmin, una forma gráfica y agradable para poder ver el proceso de creación de bases de datos y tablas:

- Comando de descarga e instalación de la interfaz gráfica de MySql: sudo apt-get install phpmyadmin –y
- Comando de configuración para abrir el fichero ejecutable del servidor Apache: sudo nano /etc/apache2/apache2.conf

Con el comando anterior se abre un fichero de configuración que copiará el siguiente código para que funcione PhpMyAdmin: "Include /etc/phpmyadmin/apache.conf". Se pega la línea de código anterior al final del fichero, después se presiona Ctrl+x, a continuación se presiona Y para aceptar los cambios realizados al fichero. Una vez realizados estos pasos, se procede a reiniciar el servidor Apache con el siguiente comando:

- Comando de reinicio del servicio Apache sin reiniciar el Raspberry Pi-3: sudo service apache2 restart

Configuración de permisos en el servidor

En Raspberry Pi-3, una vez instalado el LAMP, se necesita dar permisos de lectura y escritura a todos los archivos que se generen en este servidor, para poder ejecutar el control de la planta. A continuación, se describen los comandos que van en la consola Raspberry para otorgar los permisos a la carpeta que contiene todo el proceso de control.

- Comando que concede permisos de lectura y escritura a una carpeta completa y todos los elementos que se encuentran dentro del mismo: chmod -R 777 /var/www/

Allí se detalla el listado de los archivos con los permisos obtenidos. Si se quiere saber qué permisos se tienen antes de cambiarlos, debe ejecutarse los siguientes comandos antes que el anterior (chmod):

- Comando que lleva a la dirección establecida a través de consola: cd /var/www/html
- Comando que permite desplegar la lista de contenido del fichero dando los atributos de lectura y escritura que tienen activados a través de consola: ls –l

Algoritmos implementados

En este apartado se describe el funcionamiento de los algoritmos, tanto en el cliente-planta que tiene el sensor y actuador como en el ser-

vidor-controlador virtual. Se detallan en líneas de código paso a paso para que se ejecute adecuadamente el sistema BB. El procedimiento que se indica podría aplicarse a cualquier proceso.

Algoritmo en el cliente

El algoritmo del cliente se lo implementó en el Arduino Mega, que ya contiene la implementación del algoritmo de la planta:

1. Se incluyen las librerías para la comunicación entre el Arduino Mega y la Shield Dragino Yun para usarlos como un solo dispositivo, además de la librería para el manejo del servomotor por ángulos y la librería de intercambio de información JSON.
2. Se declara e inicializan las variables de cálculo que se van a usar durante todo el algoritmo para los cálculos de escalamiento del sensor y del disparo del ángulo en el servomotor.
3. Se define el modo de salida para los tres pines indicadores luminosos y un pin de salida PWM para ejecutar el servomotor.
4. Se inicia el proceso de bucle o lazo donde se ejecutará el dato obtenido por el sensor de distancia a través de una entrada analógica del Arduino Mega. Luego, se procede a escalar el rango de 0 a 1023 dado por el conversor ADC incorporado en el Arduino a un rango de 10 a 30 cm, que se ajusta a la realidad del sistema. Finalmente, hay que tomar en cuenta que los datos convertidos en el Arduino son enviados al servidor en formato de punto flotante, como se puede visualizar en la figura 62.
5. Una vez escalado el dato del sensor, se procede a consultar si hay información desde el servidor, mediante el comando de la librería JSON: "client.getAsynchronously ("dirección IP/dirección del proyecto/nombre del archivo.json")".
6. Se ejecuta la comprobación de conexión entre servidor y cliente a través de un contador. Si el contador llega a un límite, establece que está en desconexión y procede a dar la señal luminosa de desconexión. Además, el algoritmo predictivo entra en funcionamiento.

7. Si el servidor está disponible se procede a su conexión estableciendo la dirección IP o DNS junto con el tamaño de la cadena de caracteres. Además, para ejecutar el envío de datos se usa la función de servidor que envía a través de la sentencia PUT. Después de ejecutar esta sentencia se limpia el espacio de memoria que usa para la transferencia de datos como se muestra en la figura 63.
8. El algoritmo predictivo tiene una función independiente que en este proceso utiliza una ecuación que obedece a la tendencia del comportamiento deseado para la variable de salida (función lineal). Sin embargo, para cualquier otro proceso se recomienda utilizar como predictor la función de transferencia del controlador, pues durante la desconexión la variable de salida debe responder al controlador, debido a que se estará controlando en lazo abierto durante la desconexión.

Figura 62
Linealizar la entrada analógica

```
sensor=analogRead(0);
voltaje=(sensor*3.3)/1024.0;
Input=26.825*pow(voltaje,-1.227);
```

Fuente: los autores

Figura 63
Conexión con el servidor

```
char server[]={"http://balancin.ine4c.com/calculoPid2.php"};
char x[256];
void servidor(){
    dtostrf(Input,4,2,x);
    client.put(server,x);
    jsonBuffer.clear();
}
```

Fuente: los autores

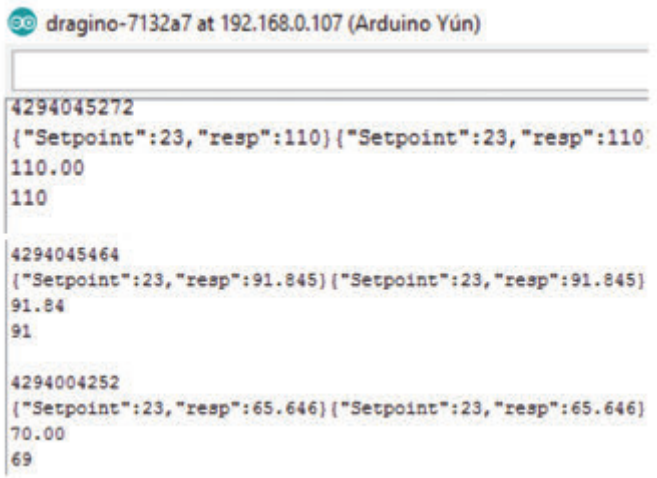
Figura 64
Algoritmo predictivo

```
void predictivo(){
  Output = (2.9695 * Setpoint) + 1.7;
  if(Output>100.0)Output = 110.0;
  else if(Output<60.0)Output = 60.0;
}
```

Fuente: los autores

Al implementar el algoritmo del cliente existe una manera de visualizar el proceso utilizando la herramienta "Monitor" propia del IDE de Arduino, la cual se muestra en la figura 65, que tiene los parámetros de envío y recepción de datos en formato JSON que se ejecutan en el cliente. Después se trata la información recibida y se muestra el ángulo de disparo que es convertido a un número entero y ejecutado por el servomotor.

Figura 65
Muestra de resultados de la planta



Fuente: los autores

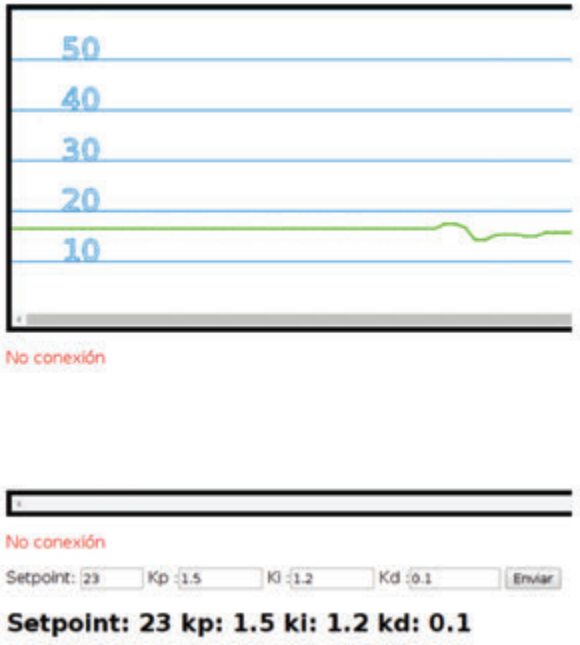
Algoritmo en el servidor

El algoritmo del servidor se lo implementó de forma local en el Raspberry Pi-3. Una vez terminado se subió el mismo algoritmo al espacio en una nube provista por un proveedor de servicio de alojamiento web (SmarterASP.NET). A continuación, se describirá brevemente la implementación del algoritmo del servidor:

1. El algoritmo del servidor se inicia buscando las librerías de gráficos que se implementan en la interfaz principal.
2. La interfaz principal tiene cinco divisiones que permiten ejecutar acciones diferentes en una sola interfaz y se inicializa dándole el tamaño, los colores y la posición de los caracteres a utilizar. En la sección de Dibujo se proporciona la información de distancia de forma gráfica. En la sección de Información se proporciona el dato numérico del sensor en forma de texto, la forma de texto de la respuesta que genera el controlador PID, los cuadros de texto para cambiar los valores constantes y las etiquetas de las constantes que se presentan en la última sección. En la figura 66 vemos la interfaz principal en modo de desconexión.
3. Se programa la interfaz gráfica de interacción con el usuario, donde se establecen el tamaño del cuadro de respuesta del sensor y el dato que recibe de la planta. A continuación, se presenta el dato del cálculo del controlador PID, el cual es mostrado en ángulo. Además, se presentan los cuadros de entrada de datos y la sección de visualización de los parámetros de configuración para PID (datos que fueron enviados al controlador como datos iniciales).
4. Una vez terminada la interfaz de interacción con el usuario, se procede a programar la implementación del algoritmo del controlador en lenguaje PHP, el cual obtiene los datos del cliente mediante la sentencia *input*. Estos datos se envían a un archivo temporal mediante formato JSON y a la vez se devuelve el dato de distancia.

5. Se ejecuta el comando: "microtime(true)*10000" que obtiene el dato de tiempo que genera la computadora cada vez que llega la información en microsegundos. Además, se procede a ejecutar varias etapas de lectura de archivos temporales que se usan para realizar los cálculos del controlador PID.
6. Una vez que se termina de ejecutar el controlador PID, se procede a abrir los archivos de escritura para guardar la información de respuesta para su envío a través del método de intercambio de información JSON. También se guardan los errores generados en los archivos temporales para usarlos posteriormente, ya que estos no se pueden guardar en espacios de memorias temporales.

Figura 66
Estado de desconexión del sistema



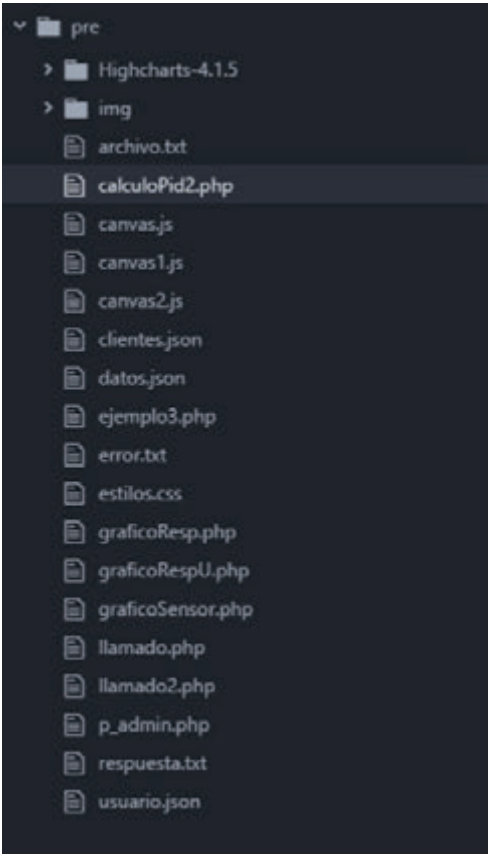
Fuente: los autores

Cuando se ejecuta el algoritmo de manera local, se guardan los datos obtenidos del sensor y los datos generados por el controlador PID en la base de datos MySql. En cambio, cuando se ejecuta el algoritmo de forma remota o en una nube no se implementa la base de datos, pues el espacio proporcionado por el proveedor de servicios no lo permite, sin embargo, si se contrata este servicio podría almacenarse igualmente.

Para describir el comportamiento de interacción de archivos que se ejecutan durante todo el proceso del controlador PID de forma global, se puede visualizar la figura 67, en donde en el archivo de extensión.php se encuentra el código de programación de interfaz de usuario y la presentación de datos gráfica y textual. También en este archivo se realiza el cálculo de la información obtenida de la planta, que en este caso nos da el sensor de distancia; esta información se obtiene de archivos temporales de extensión.txt y envía los datos de respuesta del controlador PID por medio de archivos de extensión.json.

Los archivos descritos en la figura 67 están vinculados a la interfaz principal que es "calculoPid2.php" donde las tramas de información recibida desde la planta son enviados a un archivo temporal el cual es llamado "archivo.txt" y devuelve a la interfaz principal solo el dato de distancia del sensor. Luego se realiza el cálculo del algoritmo del controlador PID, en donde se calcula el error y el error integral, estos datos son guardados en los archivos temporales "error.txt" y "errorint.txt", respectivamente, para luego utilizarlos en la función PID programada en la interfaz principal en PHP. La respuesta del controlador PID está expresada en el ángulo que va a disparar el servomotor y esta información es enviada al archivo "clientes.json" que realiza el intercambio de información con el Arduino Mega y la Shield Dragino Yun.

Figura 67
Archivos relacionados al proyecto

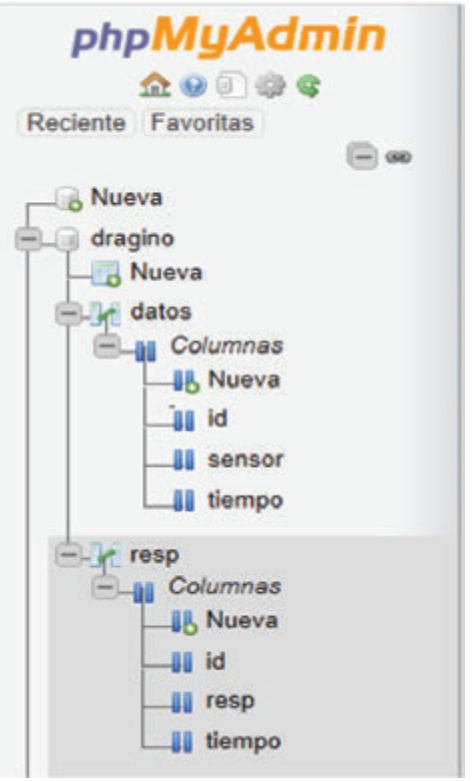


Fuente: los autores

Este proyecto tiene un componente local y uno remoto. La opción remota culmina solo con los archivos descritos anteriormente, los cuales son subidos al servidor de la nube. Para la opción local se utiliza el Raspberry Pi-3 como servidor, en donde, a más de los archivos que ejecutan el proceso descritos anteriormente, envía también los datos recibidos y calculados a una base de datos realizada en MySql, que nos

permite respaldar la información y usarla para generar un reporte que puede ser exportado e impreso como se indica en la siguiente figura:

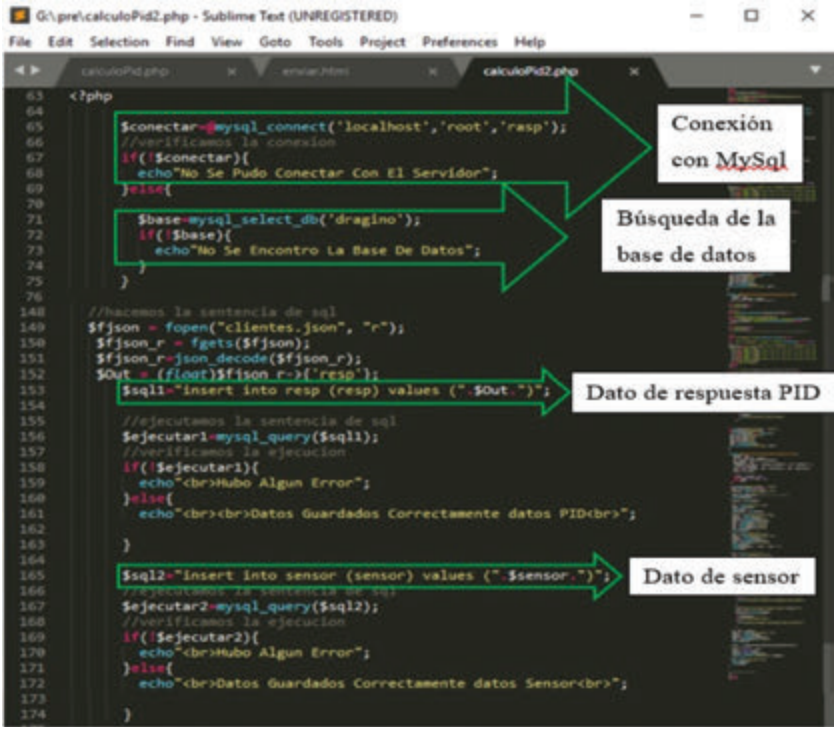
Figura 68
Base de datos MySql forma local



Fuente: los autores

La base de datos está vinculada a la interfaz principal que es "calculoPid2.php" donde usando sentencias especiales de comunicación entre PHP y MySql se envía a la base de datos la información de la distancia que llega del sensor de la planta y la respuesta del controlador PID que se va a enviar al actuador de la planta.

Figura 69
Sentencias MySql utilizadas



Fuente: los autores

En la figura 69 se muestra de forma global el código implementado para realizar el proceso de conexión y envío de datos entre PHP y MySql, el mismo que detallamos a continuación:

1. Se conecta a la base de datos MySql a través de la dirección "localhost", el usuario "root" y la contraseña "rasp".
2. Al buscar la base de datos tenemos que poner el nombre "Dragino" para que se establezca la conexión y así poder realizar la lectura y escritura de la base de datos.

3. Al tener las condiciones anteriores de conexión con MySql y PHP, además de establecer la comunicación con la base de datos Dragino se procede a escribir los datos de la respuesta del controlador PID, abriendo el archivo que contiene la información para añadirla a la tabla de Resp de la base de datos. La variable de distancia del sensor que se obtiene de la planta se añade a la tabla de Datos de la base de datos.

La información que se guardan en la base de datos es únicamente numérica, pero tiene la opción de que cada dato guardado adjunte el tiempo en el que se lo guardó, es por eso que al realizar los reportes gráficos la información es completa, tanto el dato como la fecha y hora. En el siguiente apartado se describirá el procedimiento que se utilizó para los ensayos del sistema: el ensayo de forma local y el ensayo de forma remota o en la nube.

Ensayos del sistema WNCS

Ensayo de forma local

Al realizar las pruebas de forma local, los valores tomados de la simulación sirven inicialmente para comparar los resultados del controlador virtual implementado en la Raspberry Pi3, en donde se ejecuta el algoritmo del controlador PID

No debe olvidarse que el sistema puede tener pérdidas de información en las cuales se necesita estabilizar la bola en el centro de la barra, aproximadamente, para cuando se realice la reconexión con el controlador no se reanude desde un ángulo tan inclinado sino desde un punto medio. Ahí es donde se aplica un algoritmo predictivo para equilibrar el proceso físico, algoritmo del estimador basado en el comportamiento futuro de la planta, implementado en el Ardunio Mega y para la comunicación inalámbrica en la Shiel Dragino Yun.

Ensayo de forma remota o en la nube

Al realizar las pruebas de forma remota en la nube con el proveedor de servicios SmarterASP.NET no tenemos una desconexión en la interfaz de inicio, ya que el servidor siempre tiene un dato el cual se dibuja en la respuesta al ejecutar el controlador PID. Así, la respuesta generada por la prueba local y la prueba remota de la nube son semejantes.

El sistema en la nube tiene mayor pérdida de información y para estabilizar la bola en el centro de la barra, aproximadamente, hasta realizar la reconexión con el controlador remoto se aplica un algoritmo predictivo para equilibrar el proceso

Al realizar los ensayos local y remoto se tienen varias características que destacan para cada controlador, las cuales se muestran en la siguiente tabla, así como las características singulares del sistema:

Tabla 9
Tabla comparativa del controlador local/controlador en la nube

Controlador local	Controlador remoto en la nube
El controlador se ejecuta desde un Raspberry Pi-3 que asume el rol de un servidor local.	El controlador se ejecuta desde un proveedor de servicios SmarterASP.NET que actúa como un servidor con un espacio en la nube.
La conexión cliente-planta y servidor-Raspberry Pi-3 se establece por medio de la dirección IP que se genera por el *router* de la red LAN local.	La conexión entre cliente y servidor se realiza por medio de la dirección DNS que proporciona el proveedor de servicios SmarterASP.NET, la cual es estática ya que no cambia el nombre del espacio activado.
Los archivos creados en el servidor Raspberry Pi-3 requieren tener permisos de lectura y escritura para que el controlador pueda recibir y enviar los datos a la planta.	Los archivos que usa el controlador remoto en la nube son una copia del controlador local, con la diferencia que no se necesita dar permisos de lectura y escritura para el funcionamiento del controlador PID. Además, el archivo para intercambio de información "cliente.json" debe ser creado en el servidor remoto, ya que al copiarlo del servidor local este archivo no genera la comunicación con la planta.

El porcentaje de uso del procesamiento del sistema que involucra la cantidad de información que se intercambia entre la planta y el controlador PID se puede observar en la parte superior izquierda de la pantalla de la Raspberry Pi-3 en forma de gráfica. Al ejecutar el procesamiento y el algoritmo del controlador PID, en el gráfico se puede observar que hay un alto porcentaje, lo que significa que al estar conectado con la planta el servidor requiere gran procesamiento, es decir, utiliza casi todos los recursos del Raspberry Pi-3 para su ejecución.	En el proveedor de servicios SmarterASP.NET no se tiene un indicador grafico del porcentaje de información que se intercambia entre el controlador y la planta. Al subir los archivos del controlar PID al espacio en la nube solo se especifica el ancho de banda que puede manejar el proveedor de servicios.
Al usar el dispositivo Raspberry Pi-3 los recursos del servidor son dedicados únicamente al proceso del controlador PID, sin compartir los recursos con ningún otro usuario, por ende, se tiene un servidor dedicado que genera menores tiempos de retardo en la transmisión de información.	Al usar el controlador en la nube no se tiene un proceso dedicado, sino que se utilizan recursos que son compartidos y la información recorre una mayor trayectoria por diferentes nodos de comunicación, lo cual hace que la respuesta de procesamiento del sistema sea más lenta.
La respuesta del actuador, es decir, el servomotor, es más rápida al ser ejecutado el algoritmo del controlador PID. La velocidad de reacción se debe a que usa recursos del servidor que son dedicados y los ejecuta el Raspberry Pi-3 para el intercambio de información con la planta.	La respuesta del actuador (servomotor) es más lenta con respecto al controlador local, pues aparte de tener más retardo debido al intercambio de información por la red, se producen momentos de inercia propios de elementos físicos cuando no se tiene una información continua.

Fuente: los autores

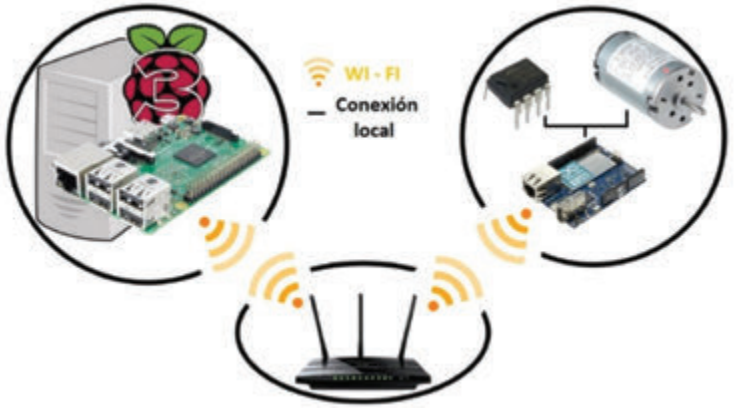
Capítulo cinco

Implementación de un WNCS para el control de voltaje de un generador DC didáctico

Descripción de la arquitectura de los dispositivos que conforman el WNCS

El sistema de control de voltaje presentado se comunica dentro de una red WAN. El cliente (sensor-actuador) está localizado en la planta, en este nodo se usa Arduino Uno + Dragino (comunicación inalámbrica). En el servidor se encuentra el controlador, implementado en un sistema embebido como es el Raspberry Pi-3. En la figura 70 se muestran los dispositivos utilizados en la implementación del sistema de control en red inalámbrica.

Figura 70
Dispositivos utilizados en la implementación del WNCS



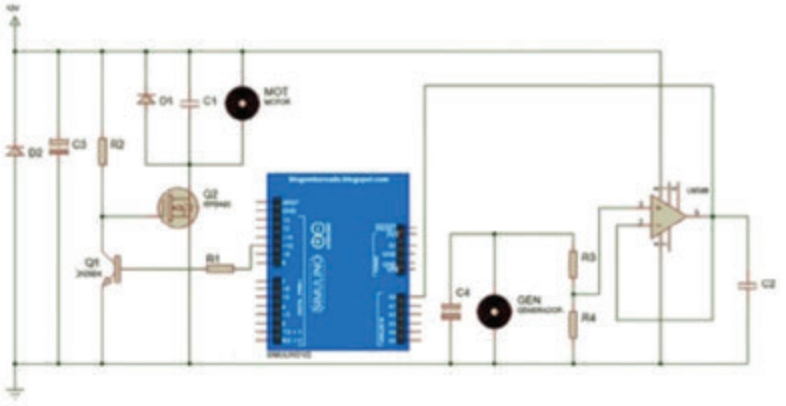
Fuente: los autores

En cuanto a los dispositivos empleados en el servidor-controlador, se puede mencionar que el controlador se encuentra albergado en la tarjeta Raspberry Pi-3 y que no se necesita ningún circuito adicional.

Figura 71
Dispositivos asociados al servidor-controlador

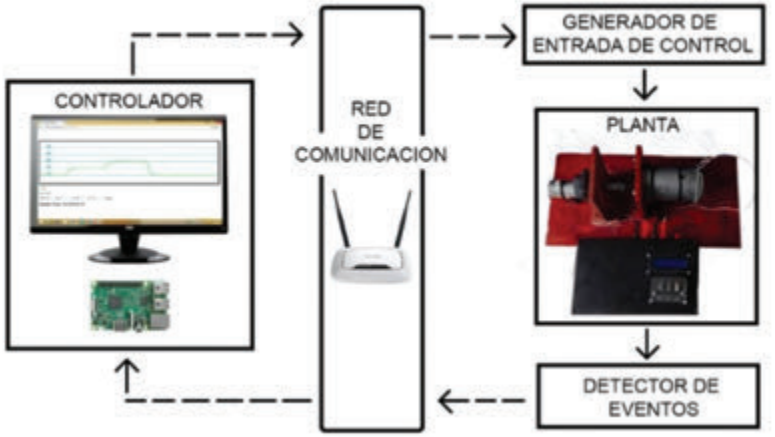
Fuente: los autores

Figura 72
Diagrama esquemático del módulo cliente (sensor-actuador)



Fuente: los autores

Figura 73
Diagrama esquemático del sistema WNCS



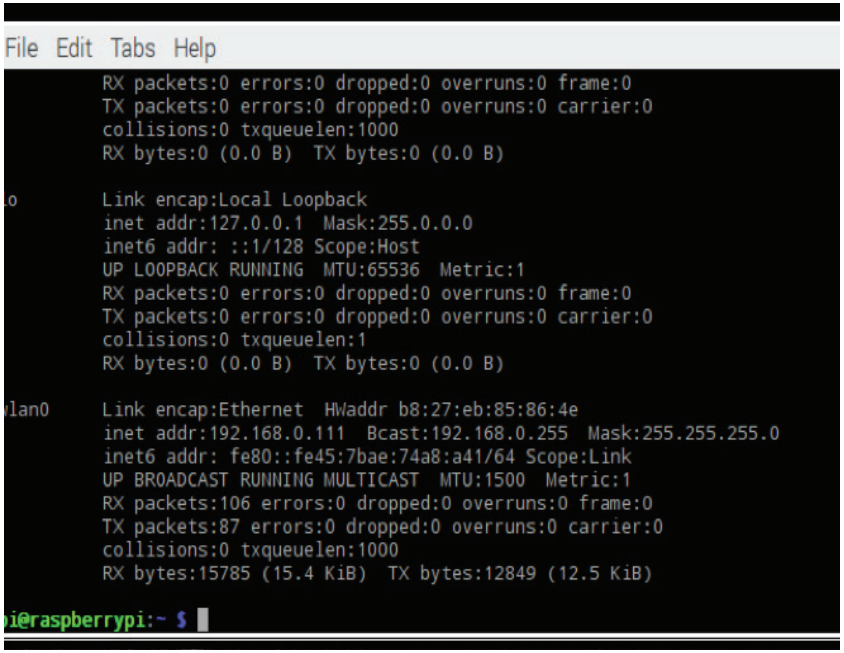
Fuente: los autores

Elementos de red

Comunicación: en este caso el encargado de comunicar los dispositivos dentro de la red inalámbrica es un *router*.

Servidor: al contar el Raspberry con conexión Wi-Fi y al enlazarlo a la red, automáticamente el *router* asigna una dirección IP para que pueda enviar y recibir paquetes de comunicación. Para verificar cuál es la dirección IP que le fue asignada, en la consola se debe ejecutar el comando "ifconfig".

Figura 74
Dirección IP Raspberry



Fuente: los autores

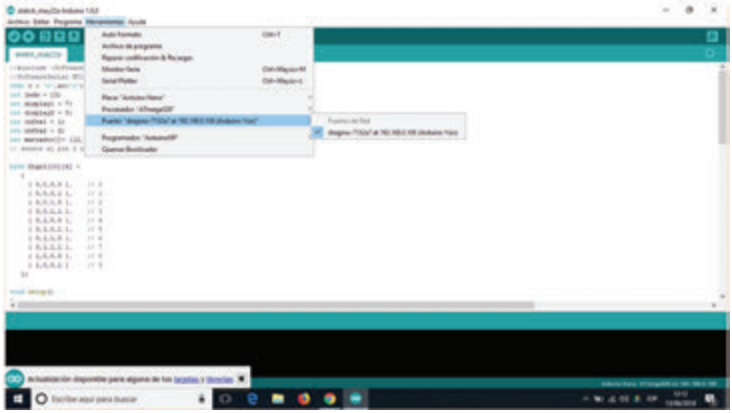
Cliente: el módulo Dragino es el encargado de proporcionar la conexión Wi-Fi en el lado del cliente. La dirección asignada a este módulo es la misma que se visualiza al momento de cargar el programa desde la plataforma Arduino IDE.

Servidor-controlador

Para el control del motor se tiene un PID, el mismo que está albergado dentro de la tarjeta Raspberry Pi-3; de igual manera se encuentran los siguientes servidores:

- Servidor WEB: recepción y presentación de datos.
- Servidor PHP: procesa y almacena los datos de la planta.
- Servidor SQL: base de datos.

Figura 75
Dirección IP Raspberry



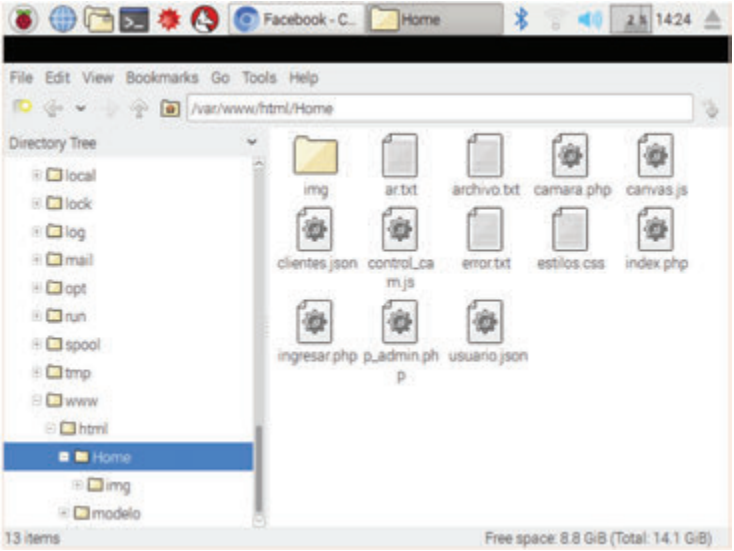
Fuente: los autores

A este servidor se puede acceder desde cualquier computador conectado a la misma red de comunicación WAN, siempre y cuando cuente con los permisos necesarios.

El controlador se encuentra en el archivo "p_admin.php", los datos se envían en formato JSON y se almacenan en el archivo "clientes.json" o en el archivo "usuarios.json":

La decodificación de los datos por parte del servidor se realiza de la siguiente manera:

Figura 76
Carpeta del servidor que contiene el controlador



Fuente: los autores

- Se accede al archivo JSON que contiene los datos.
- Se recoge el fichero de datos que se encuentra dentro del archivo.
- Se decodifica el fichero que contiene los datos.
- Se asigna cada dato a una nueva variable.

Figura 77
Decodificación de los datos formato JSON (servidor)

```
$fjson = fopen("clientes.json", "r");
$fjson_r = fgets($fjson);
$fjson_r=json_decode($fjson_r);
$Out = (float)$fjson_r->{'resp'};
$Input1 = (float)$fjson_r->{'Sensor_a1'};
$Input2 = (float)$fjson_r->{'Sensor_a2'};
fclose($fjson);
```

Fuente: los autores

La codificación de los datos sigue el proceso descrito anteriormente, es decir:

- Guardar el dato en una variable del fichero.
- Codificar los datos almacenados en el fichero.
- Guardar el fichero en el archivo deseado.

Figura 78
Codificación de los datos formato JSON (servidor)

```
$lect -> Setpoint = $set;
$lect -> resp = $resp;
$lect -> Sensor_a2 = $Input1;
$lect -> Sensor_a1 = $sensor;
$myJSON = json_encode($lect);
file_put_contents('clientes.json', $myJSON);
```

Fuente: los autores

Cliente sensor-actuador

En el cliente se encuentra conectada una tarjeta Arduino Uno + Dragino. Dragino se encargará de enviar y recibir paquetes de datos hacia el controlador, mientras que Arduino Uno se encargará de encapsular los datos del sensor para enviarlos al controlador y desencapsular los datos que envía el controlador para asignar a una salida PWM. Las librerías utilizadas en el cliente son:

Figura 79
Librerías utilizadas en el cliente[3]

```
#include <Bridge.h>
#include <HttpClient.h>
#include <ArduinoJson.h>
```

Fuente: los autores

3 Las librerías Bridge.h y HttpClient.h fueron descritas en la aplicación anterior.

Como primer paso se debe especificar la dirección del servidor. En una red local será una dirección IP, mientras que al tener un servicio de *hosting* la dirección IP cambiará por la dirección de la página web.

Figura 80
Dirección IP del servidor local

```
char server[]={"http://192.168.0.111/p_admin.php"};
```

Fuente: los autores

La librería ArduinoJson.h es la encargada de la codificación y envío hacia el servidor y viceversa, es decir, la decodificación de los datos enviados por parte del servidor. Tal como se observó en el lado del servidor, en el cliente también se debe codificar o decodificar los datos que se encuentran en formato JSON.

Para esta decodificación es necesario especificar toda la ruta de la cual se recogerán los datos y crear un vector en el cual se almacenen los datos que se obtengan del archivo del servidor (en este caso es "Datosrec"). Se utiliza una variable, en este caso *c*, para reconocer desde dónde empieza el fichero de datos, el cual comenzará con una llave ({). Finalmente, se alberga el dato del servidor en una nueva variable.

Figura 81
Decodificación de datos JSON (cliente)

```
client.get("192.168.0.111/Home/clientes.json");
                    ⋮
  char c=client.read();
  if(c=='{')lectura=1;
  if(lectura==1){x[i]=c;i++;}
 }     //Console.print(x);
JsonObject& datosrec=jsonBuffer.parseObject(x);
resp = datosrec["resp"];
Setpoint = datosrec["Setpoint"];
```

Fuente: los autores

Por su lado, para la codificación y el envío de datos hacia el servidor, primero, se lleva a cabo la medición y el escalamiento de la variable (voltaje del generador), luego se realiza la conversión del dato de la variable (dtostrf(variable, 4, 2, nueva_variable)) y finalmente se envía la variable.

Figura 82
Codificación de datos JSON (cliente)

```
Input = analogRead(0)*vmax/1023;
        ⋮
dtostrf(Input,4,2,x);
client.put(server,x);
```

Fuente: los autores

En la figura 82, "server" hace referencia a la dirección del servidor, la cual se detalló en la figura 80.

Por parte del cliente se tienen dos etapas:

- Sensor. El valor medido en la planta pasa por un proceso de acondicionamiento de señal, luego es enviado al controlador y comparado con el *set point* especificado. El error presente en la planta es procesado en un algoritmo PID discreto en el servidor, el cual retorna un valor comprendido entre 0 y 255.
- Actuador. El valor entre 0 y 255 será asignado a una salida PWM en el Dragino, que a su vez provocará la conmutación de un Mosfet encargado de controlar al motor.

En la figura 83 se muestra el diagrama de cómo se realiza la comunicación entre el cliente y el servidor.

Figura 83
Diagrama de comunicación cliente-servidor

Fuente: los autores

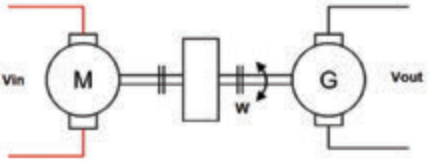
El formato JSON ordena los datos y permite extraerlos fácilmente, sin métodos de discriminación. Para esto el formato JSON tiene la siguiente forma:

{“sensor”:0,”error”:10,”resp”:0,”Out_a1”:0,”Sensor_a1”:0,”SP”:10,”Sensor_a2”:0}

El modelo del sistema del control de voltaje de un motor-generador DC

El sistema de control de voltaje aquí presentado consta de dos motores acoplados entre sí, donde se controla la velocidad de uno de ellos para obtener un voltaje de salida determinado en los terminales del segundo motor, actuando como generador:

Figura 84
Diagrama del sistema controlador de voltaje DC



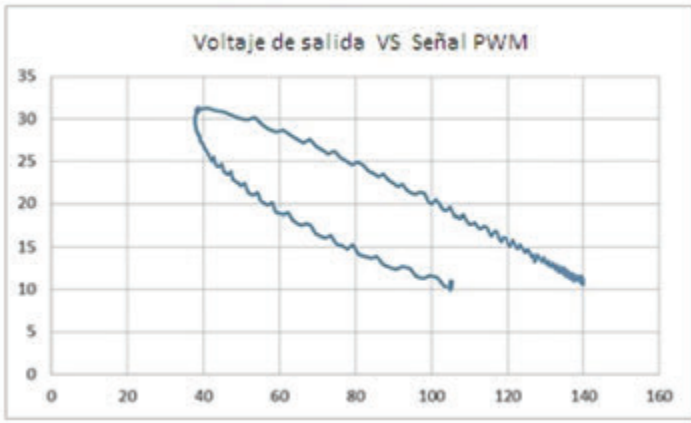
Fuente: los autores

Metodología para la simulación e implementación

Para la obtención del modelo matemático se deben tomar muestras de entrada y salida de la planta con un tiempo de muestreo de 10 ms y un retardo en el comportamiento de la planta de 450 ms; por su lado, en el generador se tiene un voltaje comprendido de 10 a 30 voltios.

Los datos obtenidos se almacenan en una tabla de Excel, donde se comprueba la validez de los datos mediante graficas de la entrada y salida.

Figura 85
Gráfica de la muestra voltaje de salida vs señal PWM



Fuente: los autores

En la figura 85 se observan los datos obtenidos al iniciar las mediciones partiendo desde 10 V. La señal PWM inicia en un valor aproximado de 115, llegando a 30V con un valor en la señal PWM aproximado de 38; en este tramo de la gráfica el motor está rompiendo la inercia del reposo. Una vez que se encuentra en funcionamiento, el comportamiento del sistema es como el que se muestra en el segundo tramo de la gráfica. Se recomienda tomar tres o más muestras de datos para comparar cuál es la que más se aproxima al modelo del sistema.

Los datos obtenidos se deben importar al software Matlab, con la ayuda de la herramienta IDENT. Luego se determina la función de transferencia de cada gráfica con el propósito de obtener aquella que se apegue de mejor manera al sistema.

$$\frac{V}{pwm} = \frac{0{,}5537}{s^2 + 0{,}5606s + 2{,}355} \tag{5.1}$$

La función de transferencia es discretizada con el comando "c2d" de Matlab:

$$\frac{V}{pwm} = \frac{0{,}00002763z^{-1} + 0{,}00002758z^{-2}}{1 - 1{,}994z^{-1} + 0{,}9944z^{-2}} \tag{5.2}$$

Con la función discretizada se obtiene la ecuación en diferencias y despejando V se tiene lo siguiente:

$$V = 1{,}994\, V_{i\text{-}1} - 0{,}9944\, V_{i\text{-}2} - 0{,}00002763\, PWM_{i\text{-}1} + 0{,}00002758\, PWM_{i\text{-}2} \tag{5.3}$$

Esta ecuación es el estimador de la planta, que será usado por el controlador para estimar la respuesta de la planta real.

Diseño del controlador PID predictivo para la planta

En la planta, la variable que se desea controlar es el voltaje de salida en los terminales del generador. Para esto se debe manipular el voltaje de entrada del motor y para el control del motor se debe tomar en cuenta los siguientes puntos:

- El voltaje inicial para que el que el motor logre romper la inercia.
- El acople de los ejes debe ser lo más exacto posible, para evitar que el desbalance produzca picos de voltaje.

Por la naturaleza del motor, cuando gira el rotor el campo magnético producido aumenta, lo cual provoca que su velocidad también aumente.

Algoritmo predictivo para el control PID basado en eventos para el sistema

El razón para utilizar un controlador predictivo se presenta al existir pérdida de comunicación, pues su función principal es proveer de soporte al sistema y entregar información similar a la real hasta que la comunicación se restablezca. La pérdida de comunicación puede estar determinada por los siguientes casos: el valor de error entre el *set point* y el voltaje de salida de la planta es prácticamente nulo o hay retardo en los paquetes de comunicación.

La figura 86 muestra la función en donde se realiza el control PID por parte del servidor, limitando los valores máximos y mínimos para el correcto funcionamiento del sistema.

Figura 86
Control PID en el servidor

```
function pid($Setpoint,$Input,$Out,$kp,$ki,$kd,$Input1,$Input2){
  $dt = 0.01;
  $error = $Setpoint - $Input;
  $k1 = $kp + $ki * $dt + $kd/$dt ;
  $k2 = $ki*$dt-2*$kd/$dt;
  $k3 = ($kd/$dt)-$kp ;
  $Out = $Out-($k1*($Setpoint-$Input)+$k2*($Setpoint-$Input1)+$k3*($Setpoint-$Input2));

   if($Out > 160)$Out = 160;
   else if($Out < 0) $Out = 0;
   return $Out;
  }
```

Fuente: los autores

En los requerimientos, se debe anotar que el algoritmo PID necesita para su ejecución los dos últimos datos de voltaje de salida obteni-

dos por el sensor, así como los dos últimos datos de PWM emitidos por el controlador; esto con el propósito de calcular cuál es la salida óptima y asignarla al motor.

Cuando el sistema se encuentra estabilizado, el controlador predictivo toma el control del sistema cortando la comunicación y estimando la salida, la que debería encontrarse en ese estado, asignándola al controlador. De la misma manera, al existir retardo en los paquetes de comunicación, el controlador predictivo estima el valor que se debe asignar al actuador.

Algoritmos implementados en el WNCS

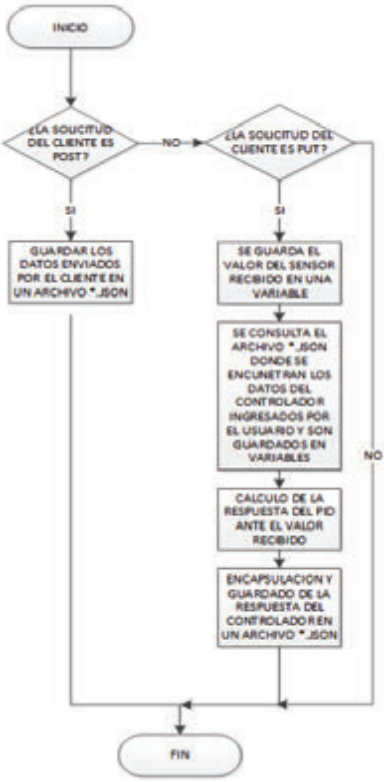
Para el sistema presentado se utilizarán dos algoritmos, los cuales deben determinar el estado actual de la planta para de esta manera mantener el control con el servidor o delegar el control al controlador predictivo.

Algoritmo en el servidor-controlador predictivo PID asíncrono

Cuando el usuario cambia los valores de *SP*, K_p, K_i o K_d envía una solicitud POST. Ante esta solicitud, el servidor únicamente guardará esos datos en un archivo para consultarlo posteriormente. Si la solicitud es PUT, significa que son datos enviados por el cliente (Arduino), entonces los datos recibidos por el servidor (dato del sensor y errores anteriores) son decodificados y guardados en variables, al mismo tiempo que se consultan los datos guardados en el archivo de la solicitud POST. Con esto se tiene *SP*, K_p, K_i, K_d, los errores anteriores y el valor del sensado en la planta. Estas variables son necesarias para el cálculo del PID.

Una vez calculada la respuesta del PID se guarda la respuesta (valor PWM) y los dos últimos errores (*SP* = valor sensado) de la planta, en otro archivo, el cual el cliente puede consultar para obtener la respuesta del servidor.

Figura 87
Algoritmo implementado en el servidor



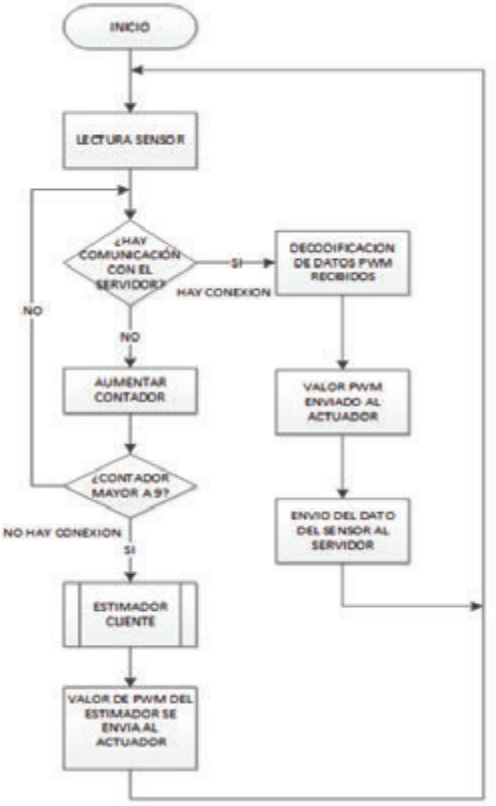
Fuente: los autores

Algoritmos en el cliente sensor-actuador

En el cliente el algoritmo se desarrolla con los siguientes pasos: medición del voltaje en el generador (V-out) y conexión con el servidor. Para la conexión existen dos posibilidades: la primera es que al establecerse la conexión, solicita la respuesta del controlador (valor PWM), la decodifica, envía el valor del PWM al actuador y por último envía el valor sensado

inicialmente al servidor para un nuevo cálculo. La segunda es que si la conexión con el servidor falla se incrementa el contador y lo vuelve a intentar, esta operación será realizada nueve veces (el valor de nueve es un valor escogido empíricamente en el cual se espera la conexión con el servidor por un tiempo prudente), una vez que el contador es mayor a nueve considera que la conexión con el servidor ha fallado y usa el estimador del cliente para controlar la planta y el proceso empieza nuevamente.

Figura 88
Algoritmo implementado en la planta



Fuente: los autores

La figura 89 detalla la función del control PID predictivo en el lado del cliente. De igual manera se establecen los valores máximos y mínimos para que el sistema funcione correctamente.

Figura 89
Control PID predictivo en el cliente

```
void loop() {
  cont = 0;
  tim=micros();
  V2 = V1; V1=V;
                    ⋮
 void predictivo(){
   V = -0.2318*PWM + 44.327;
   PWM = PWM1 -1.641*(Setpoint-V1)+0.6786*(Setpoint-V2);
     if(PWM>141)PWM = 141;
   else if(PWM<0)PWM = 0;
   }
```

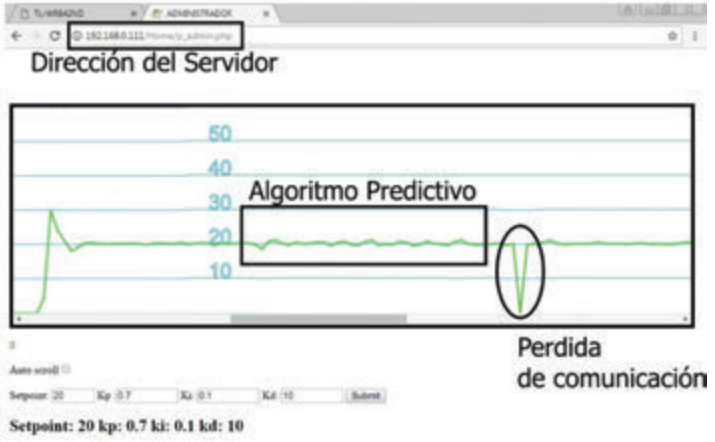
Fuente: los autores

Resultados de la aplicación

Las figuras 90 y 91 muestran los resultados del sistema implementado, centrándose en el controlador. Se puede apreciar el comportamiento del sistema en diferentes condiciones:

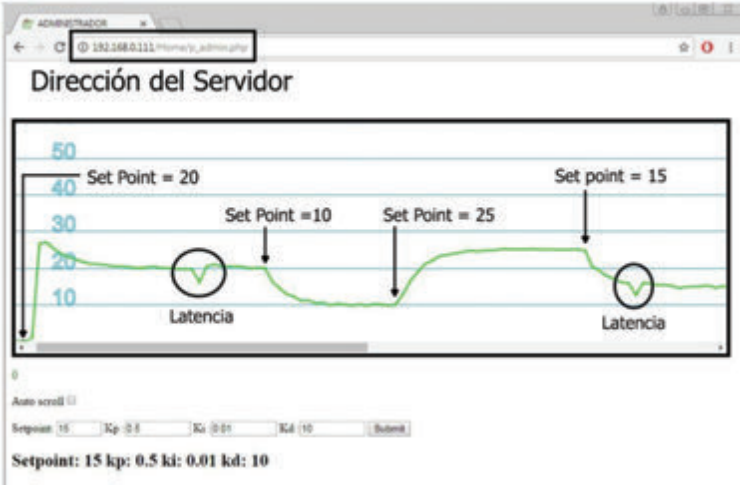
- Enlazado con el servidor (controlador PID).
- Al existir latencias en la red.
- Perturbaciones.
- Cambios de *SP*.
- Al activarse el control predictivo.

Figura 90
Algoritmo implementado en el controlador 1



Fuente: los autores

Figura 91
Algoritmo implementado en el controlador 2



Fuente: los autores

En la figura 91 el sistema parte del reposo. Al enviar un *SP* de 20 V se presenta un sobrepico hasta alcanzar la estabilidad. Ciertas latencias que se presentan en la comunicación hacen que el control varíe. Al reanudarse la comunicación se vuelve a estabilizar en el *SP* establecido y posterior a esto se presentan varios cambios de *SP* en los que se puede ver que el sistema responde de manera adecuada.

Anexos[4]

Anexo A
Modelo matemático de la planta BB

Para obtener el modelo matemático que describe el comportamiento de la planta, es decir, la función de transferencia de la planta, se realizó un proceso experimental que iremos describiendo a continuación.

Primero se implementó la planta con el controlador PID incluido, en el cual se configuraron los parámetros de una forma empírica experimental siguiendo los criterios de estabilización de Zigler-Nichols. Luego de lograrse estabilizar adecuadamente, se tomó los datos, muestras de la entrada y de la salida del sistema (vale recalcar que tal experimento para la toma de datos se lo hizo de manera local, es decir, todos los elementos del sistema BB: sensor-actuador, planta y controlador, estaban físicamente unidos).

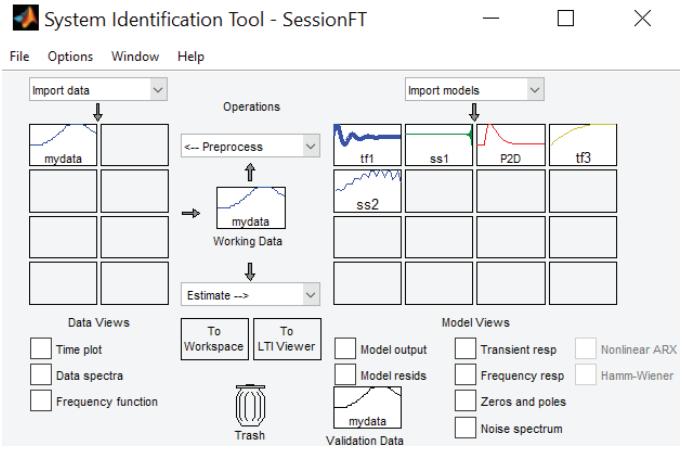
Los datos de salida del sistema se obtuvieron de las mediciones, que eran impresas por medio de Arduino mediante comunicación serial. A continuación se copió estos datos en una hoja de Excel para realizar una tabla de datos, tanto de entrada como de salida (archivo: "funcion_pid_bloques.xls") y luego se tomó esos datos en Matlab mediante el *scrip*: FunctionTrans.m

4 Para la información de los algoritmos implementados, tanto para el sistema Beam-Ball como para el sistema del motor-generador, así como ejemplos de comunicación Wi-Fi para la implementación de WNCS, comuníquese con el autor al correo: cpillajo@ups.edu.ec o capillajo@gmail.com.

```
% PARA IDENTIFICAR LA FUNCION DE TRANSFERENCIA
% teniendo datos tomados de las mediciones reales.
% estos datos estan en un archivo de excel
% primero leemos el rango de datos que se desea de
% la funcion de excel
clear all; close;
datosuy=xlsread('funcion_pid_bloques.xlsx','Hoja1','B48:C135');
u=datosuy(:,2);
y=datosuy(:,1);
ident
```

Mediante la herramienta IDENT, se realizó un ajuste de los datos para ver cuál era la función de transferencia que más se acoplaba a los datos tomados experimentalmente. Siendo *u* los datos de entrada y los datos de salida *y*, a un tiempo de muestreo $T = 0{,}01$ s, como se puede observar, se realizaron algunas aproximaciones, obteniéndose la tf1 como la función de transferencia adecuada, con 2 polos y 1 cero sin retardo en tiempo continuo.

Figura 92
Herramienta IDENT para sacar FT



Fuente: los autores

Posteriormente, con estos parámetros se empezó a realizar la simulación:

```
>> tf1
tf1 =
        From input “u1” to output “y1”:
        1.068 s + 151.3
        ----------------------
        s^2 + 5.458 s + 163.9
Name: tf1
Continuous-time identified transfer function.
Parameterization:
        Number of poles: 2 Number of zeros: 1
        Number of free coefficients: 4
        Use “tfdata”, “getpvec”, “getcov” for parameters and their uncertainties.
Status:
Estimated using TFEST on time domain data “mydata”.
Fit to estimation data: 76.63% (simulation focus)
FPE: 0.8398, MSE: 0.7475
```

Para corroborar el anterior proceso se utilizó también el criterio de la función de transferencia del proceso:

```
P2U =
Process model with transfer function:
                        Kp
        G(s) = ---------------------
                1+2*Zeta*Tw*s+(Tw*s)^2
        Kp = 0.92319
        Tw = 0.3907
        Zeta = 0.21306
Name: P2U
Parameterization:
        ‘P2U’
        Number of free coefficients: 3
        Use “getpvec”, “getcov” for parameters and their uncertainties.
Status:
Estimated using PROCEST on time domain data “mydata”.
Fit to estimation data: 76.63% (prediction focus)
FPE: 0.823, MSE: 0.7476
```

Para el modelamiento se identificó las respuestas del sistema en conjunto para lo cual se midió la señal de salida del sistema como es la distancia de la bola al eje respecto a una señal de referencia, estos datos fueron utilizados para

la identificación del sistema utilizando el Toolbox de Matlab (IDENT), De los modelos identificados, se seleccionó la función de transferencia, que representa mejor el comportamiento del sistema, así como la función del PID tomada de datos experimentales con las contantes $K_p = 8$, $K_i = 0{,}5$ y $K_d = 2{,}8$, se sabe que la función del controlador es $PID = \left(K_p + sK_d + \frac{K_i}{s}\right)$, por lo tanto, $Gt = PID * FTp$:

$$Gt = \frac{6{,}049}{s^2 + 1{,}091s + 6{,}552};$$

$$PID = \frac{2{,}8s^2 + 8s + 0{,}5}{s};$$

$$FTp = \frac{6{,}049s}{\left((s^2 + 1{,}091s + 0{,}503) * (2{,}8s^2 + 8s + 0{,}5)\right)} \quad \text{(A1)}$$

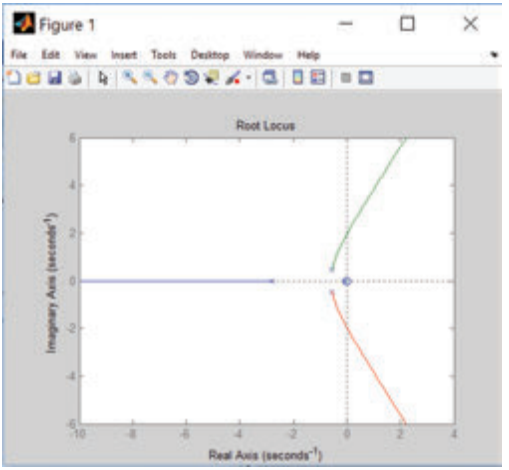
Así, se ejecuta el siguiente *script* en Matlab:

```
nump=[6.049 0];denp=[2.8 11.05 10.64 4.5695 0.2515]; % CON LA FUNCION
IDENT
gpt=tf(nump,denp) % Función de transferencia del proceso
[p,z]=pzmap(gpt)
rlocus(gpt) % Grafico de Polos y Ceros de la FT del proceso sin controlador
[A,B,C,D]=tf2ss(nump,denp)
error=0;
dt=0.01; Tmax=5;
t=0:dt:Tmax;
s=tf('s');
Planta=tf(nump,denp);
Kp=8; Ki=0.5;Kd=2.8;
Cpid= pid(Kp,Ki,Kd)
%Sistema lazo cerrado realimentación Unitaria
sys_c1=feedback(Cpid*Planta,1)
[p,z]=pzmap(sys_c1)
rlocus(sys_c1)
title(sprintf('LGR LazoCerrado, Kp=%f, Ki=%f, Kd=%f',Kp,Ki,Kd));
step(sys_c1);
        m= step(sys_c1,t);
        % %Calculo del error
        error=1-m;
        IAE=trapz(t,abs(error))
```

Entonces, se obtienen los siguientes resultados:

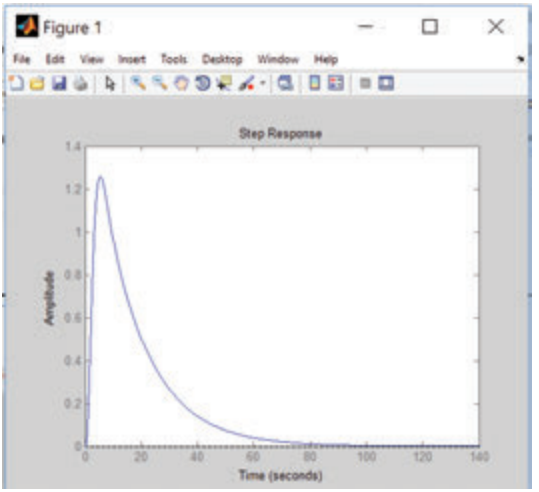
```
gpt =
        6.049 s
  ----------------------------------------------------
  2.8 s^4 + 11.05 s^3 + 10.64 s^2 + 4.569 s + 0.2515

Continuous-time transfer function.
p =
  -2.7899 + 0.0000i
  -0.5463 + 0.4529i
  -0.5463 - 0.4529i
  -0.0639 + 0.0000i
z =
        0
Matrices de estado de la FT del proceso
A =
    -3.9464   -3.8000   -1.6320   -0.0898
     1.0000         0         0         0
          0    1.0000         0         0
          0         0    1.0000         0
B =
     1
     0
     0
     0
C =
     0     0   2.1604         0
D =
     0
```

Figura 93
Gráfico de polos y ceros de la FT sin controlador



Fuente: los autores

Figura 94
Repuesta paso de la planta sin controlador

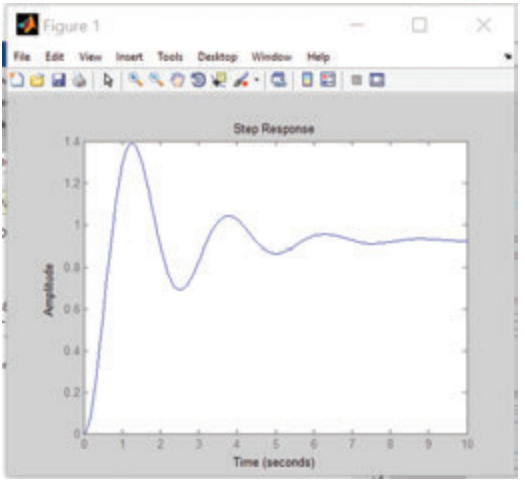


Fuente: los autores

Ahora se programa el sistema con controlador y realimentación unitaria:

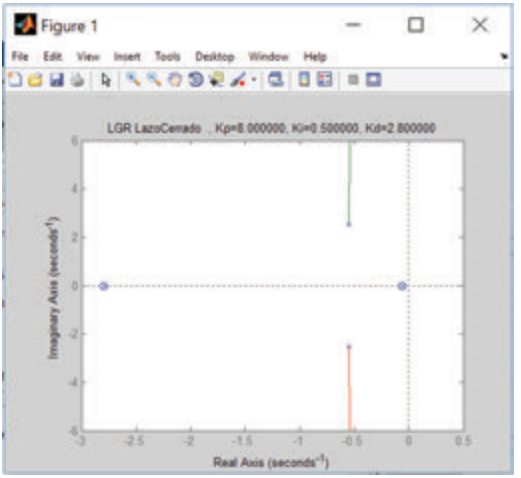
```
sys_c1 =
          16.94 s^3 + 48.39 s^2 + 3.025 s
          ---------------------------------------------------------
          2.8 s^5 + 11.05 s^4 + 27.58 s^3 + 52.96 s^2 + 3.276 s

Continuous-time transfer function.
Polos y Ceros de la FT del proceso, con controlador y realimentación unitaria
p =
          0.0000 + 0.0000i
          -2.7917 + 0.0000i
          -0.5454 + 2.5016i
          -0.5454 - 2.5016i
          -0.0639 + 0.0000i
z =
          0
          -2.7932
          -0.0639
IAE =
          1.1134
```

Figura 95
Respuesta a una señal paso del sistema con PID



Fuente: los autores

Figura 96
LGR de la planta con controlador PID continúo



Fuente: los autores

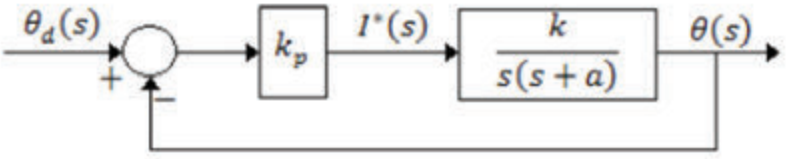
Modelo discreto de la planta BB

También se aplicó otro método en base a los datos medidos y en función de ciertos parámetros necesarios, en donde se obtuvo la función de transferencia del sistema motor y luego del sistema bola.

Modelo del motor

Se consideró el sistema motor mediante el siguiente modelo esquemático:

Figura 97
Control proporcional de posición para la riel



Fuente: los autores

Donde la función de transferencia correspondiente es:

$$\frac{\theta(s)}{\theta_d(s)} = \frac{k_p k}{s^2 + as + k_p k} = \frac{w_n^2}{s^2 + 2\zeta w_n s + w_n^2} \qquad \text{(A2)}$$

Tal que:

$$k = \frac{w_n^2}{k_p}\,,\quad a = 2\zeta w_n\,,\quad W_n = \sqrt{k_p k}$$

Donde W_n se conoce como la frecuencia natural y ζ es un factor de amortiguamiento.

$$\zeta = \sqrt{\frac{ln^2\left(\frac{M_p(\%)}{100}\right)}{ln^2\left(\frac{M_p(\%)}{100}\right)+\pi^2}} \tag{A3}$$

$$w_d = \frac{1}{t_r}\left[\pi - tan^{-1}\left(\frac{\sqrt{1-\zeta^2}}{\zeta}\right)\right] \tag{A4}$$

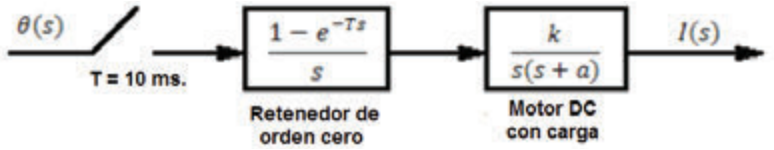
$$w_n = \frac{w_d}{\sqrt{1-\zeta^2}} \tag{A5}$$

Con estos datos medidos se obtuvo:

Mp = 0,09
Tr = 1,17
Kp = 5,9
K = 0,972862709
A = 2,914900359

Entonces, discretizando la función y considerando:

Figura 98
Diagrama de la función de transferencia de un motor DC



Fuente: los autores

Del modelo anterior se tiene:

$$G1(s) = \frac{\theta(s)}{I(s)} = \frac{1-e^{-Ts}}{s}\,\frac{k}{s(s+a)} \tag{A6}$$

A este modelo se le aplica la transformada *z* y entonces se obtiene:

$$G_1(z) = \frac{\theta(z)}{I(z)} = Z\left[(1 - e^{-Ts})\frac{k}{s^2(s+a)}\right] \quad \text{(A7)}$$

$$G_1(z) = \frac{\theta(z)}{I(z)} = Z\left[(1 - e^{-Ts})\right] Z\left[\frac{k}{s^2(s+a)}\right] \quad \text{(A8)}$$

Donde:

$$A = -\frac{k}{a^2} \;; \quad B = \frac{k}{a} \;, \quad C = \frac{k}{a^2}$$

Si:

$$Z[e^{Ts}] = z \;,\; Z\left[\frac{1}{s}\right] = \frac{1}{1 - z^{-1}} \;,\; Z\left[\frac{1}{s^2}\right] = \frac{Tz^{-1}}{(1 - z^{-1})^2} \;,\; Z\left[\frac{1}{(s+a)}\right] = \frac{1}{1 - e^{-aT}z^{-1}}$$

$$G_1(z) = (1 - z^{-1})\left(\frac{A}{1 - z^{-1}} + \frac{BTz^{-1}}{(1 - z^{-1})^2} + \frac{C}{1 - e^{-aT}z^{-1}}\right) \quad \text{(A9)}$$

Reemplazando los datos obtenidos anteriormente se tienen los parámetros $k = 0{,}972$, $a = 2{,}914$ del sistema BB y tiempo de muestreo $T = 10$ ms. Así, tenemos que A = -0,1144, B = 0,3337 y C = 0,1144. Por lo tanto:

$$G_1(z) = \frac{\theta(z)}{I(z)} = \frac{0{,}0004253\, z^{-1} + 0{,}0041308\, z^{-2}}{1 - 1{,}916267 z^{-1} + 0{,}916267 z^{-2}} \quad \text{(A10)}$$

La ecuación (A9) es la función de transferencia en transformada *z* del modelo del motor DC. A continuación, presentamos la función discreta del modelo anterior para su programación directa.

$$(1 - 1{,}916267 z^{-1} + 0{,}916267 z^{-2})\theta(z) = (0{,}0004253\, z^{-1} + 0{,}0041308\, z^{-2})I(z) \quad \text{(A11)}$$

Aplicando la transformada *z*:

$$x(n+k) = z^{-k}X(z) \tag{A12}$$

Se tiene:

$$\theta(n) = 1.916267\,\theta(n-1) - 0.916267\,\theta(n-2) + 0.0004253\,I(n-1) + 0.0041308\,I(n-2) \tag{A13}$$

La ecuación (A12) es la función que representa al modelo dinámico del motor DC y es el que se programa en cualquier sistema digital.

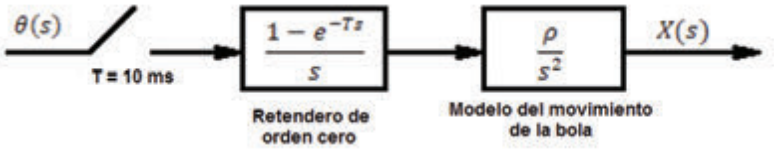
Modelo de la bola

El modelo dinámico de la bola consiste en una bola que gira sobre un riel donde:

$$\frac{X(s)}{\theta(s)} = \frac{\rho}{s^2} \tag{A14}$$

Para su conversión en la transformada *z* se diseña el siguiente modelo:

Figura 99
Diagrama del modelo de la bola con retenedor de orden cero



Fuente: los autores

Del diagrama anterior se tiene:

$$G_2(s) = \frac{X(s)}{\theta(s)} = \frac{1 - e^{-Ts}}{s}\,\frac{\rho}{e^2} \tag{A15}$$

Aplicando la transformada *z* a la ecuación anterior, se tiene:

$$G_2(z) = \frac{X(z)}{\theta(z)} = Z\left[(1 - e^{-Ts})\left(\frac{\rho}{e^3}\right)\right] \quad \text{(A16)}$$

$$G_2(z) = Z[(1 - e^{-Ts})]\ Z\left[\left(A\frac{2}{e^3}\right)\right] \quad \text{(A17)}$$

Donde $A = \rho/2$, si:

$$Z[e^{Ts}] = z \quad \text{y} \quad Z\left[\frac{2}{s^3}\right] = \frac{T^2 z^{-1}(1 + z^{-1})}{(1 - z^{-1})^3} \quad \text{(A18)}$$

Entonces:

$$G_2(z) = \frac{AT^2 z^{-1} + AT^2 z^{-2}}{1 - 2z^{-1} + z^{-2}} \quad \text{(A19)}$$

Sustituyendo el parámetro Rho = 3 del BB, entontes A = 1,5; y si el tiempo de muestreo de T = 10 ms, entonces:

$$G_2(s) = \frac{3}{s^2} \quad \text{(A20)}$$

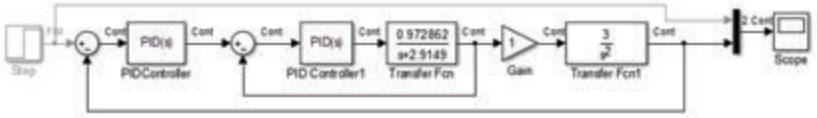
$$G_2(z) = \frac{X(z)}{\theta(z)} = \frac{0{,}00015z^{-1} + 0{,}00015z^{-2}}{1 - 2z^{-1} + z^{-2}} \quad \text{(A21)}$$

Para la programación directa utilizando la propiedad de la transformada *z* en la ecuación (A20), se tiene:

$$x(n) = 2\,x(n-1) - x(n-2) + 0{,}00015\,\theta(n-1) + 0{,}00015\,\theta(n-2) \quad \text{(A22)}$$

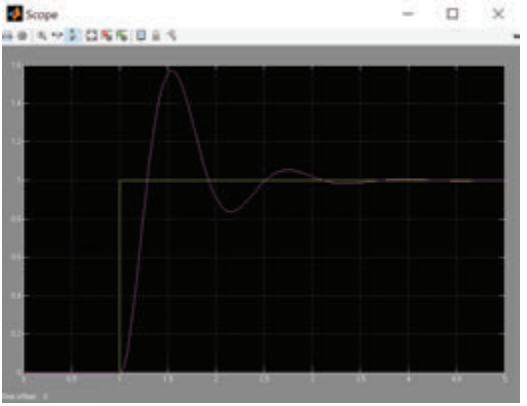
La ecuación (A21) es la función que representa al modelo dinámico del movimiento de la bola y es el que se programa en cualquier sistema digital. Realizando la respectiva simulación en Simulink de las funciones de transferencia se tiene:

Figura 100
Diagrama de bloques en Simulink del sistema BB



Fuente: los autores

Figura 101
Respuesta del sistema BB ante una entrada paso en Simulik



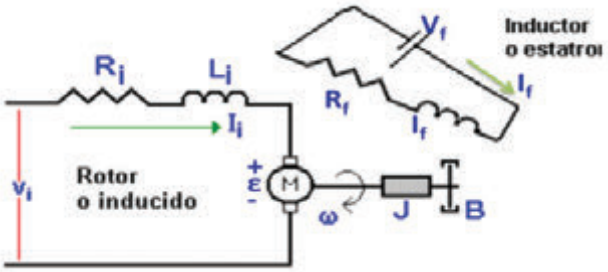
Fuente: los autores

Anexo B
Modelo matemático del sistema motor-generador

Modelo del motor DC

Un motor DC consta de rotor y estator, donde el campo generado por el estator es creado por la corriente que pasa por las bobinas del mismo, mientras el rotor (inducido) es la parte giratoria del motor. De esta configuración se saca el esquema del motor presentado en la siguiente figura:

Figura 102
Esquema motor DC



Fuente: los autores

Donde:

Rf = resistencia bobinado del estator.
Lf = inductancia bobinado del estator.
If = corriente en el bobinado del estator.
Vf = voltaje alimentación del estator.
Ri = resistencia del bobinado del rotor.
Li = inductancia del bobinado del rotor.
Ii = corriente en el bobinado del rotor.
Vi = voltaje de alimentación del rotor.
W = velocidad angular.
J = momento de inercia.
B = coeficiente de rozamiento viscoso.

El modelado matemático del motor DC requiere de una ecuación mecánica y una eléctrica que se basan en las leyes de la dinámica y las leyes de Kirchhoff. La ecuación mecánica modela el movimiento del rotor, mientras que la ecuación eléctrica modela el circuito eléctrico del inducido. Al aplicar una tensión Vi al inducido, circula por él una corriente Ii, y debido a esta corriente, por el rotor, se inducirá una fuerza contra-electromotriz (ley de Lenz: "toda corriente se opone a la causa que la produce") cuyo valor vendrá determinado por la expresión:

$$\varepsilon = k_b \times \omega(t) \tag{B1}$$

Donde k_b es el coeficiente de fuerza contra-electromotriz. Así, aplicando la segunda ley de Kirchhoff la tensión es igual a:

$$V_i = V_{R_i} + V_{L_i} + \varepsilon \tag{B2}$$

Despejando la tensión útil y aplicando ley de OHM, se tiene:

$$V_i - \varepsilon = R_i \times I_i(t) + L_i \times \frac{dI_i(t)}{dt} \tag{B3}$$

Reemplazando (B1) en (B3), tenemos:

$$V_i - k_b \times \omega(t) = R_i \times I_i(t) + L_i \times \frac{dI_i(t)}{dt} \tag{B4}$$

El rotor realizará su movimiento debido al torque electromagnético τ_e generado por el campo magnético que se produce en el estator, a su vez, este dependerá de la corriente que circula en la armadura. De esta manera la ecuación es:

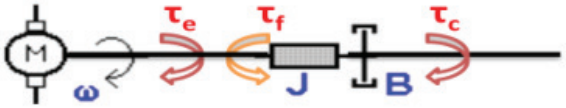
$$\tau_e = k_p \times I_i(t) \tag{B5}$$

Donde k_p es la constante de torque electromagnético.

El motor en su movimiento giratorio arrastra una carga, creándose un par-motor resultante τ_e, a su vez, se tiene también fricción en el sistema que

depende de la velocidad a la cual gira el rotor y este causa un torque τ_f que va en sentido opuesto al movimiento. Obsérvese esto en la siguiente figura:

Figura 103
Diagrama de torques en el rotor



Fuente: los autores

El símbolo α representa la aceleración angular, donde:

$$\alpha = \frac{d\omega(t)}{dt} \tag{B6}$$

La ecuación que describe a τ_c es:

$$\tau_c = J \times \alpha = J \times \frac{d\omega(t)}{dt} \tag{B7}$$

La ecuación que describe a τ_f es:

$$\tau_f = B \times \omega(t) \tag{B8}$$

Realizando una sumatoria de torques se tiene:

$$\sum \tau = J \times \alpha \tag{B9}$$

$$\tau_e - \tau_f = \tau_c \tag{B10}$$

Reemplazando (B5), (B7) y (B8) en (B10) y despejando τ_e, se tiene:

$$k_p \times I_i(t) - B \times \omega(t) = J \times \frac{d\omega(t)}{dt} \tag{B11}$$

$$k_p \times I_i(t) = J \times \frac{d\omega(t)}{dt} + B \times \omega(t) \tag{B12}$$

Despejando $I_i(t)$ de (B12) y luego derivando con respecto al tiempo, se tiene como resultado:

$$I_i(t) = \frac{J \times \frac{d\omega(t)}{dt} + B \times \omega(t)}{k_p} \tag{B13}$$

$$\frac{dI_i(t)}{dt} = \frac{J \times \frac{d\omega^2(t)}{dt} + B \times \frac{\omega(t)}{dt}}{k_p} \tag{B14}$$

Sustituyendo (B13) y (B14) en (B4), quedará una ecuación diferencial de segundo orden (aparece la segunda derivada), no homogénea, lineal y de coeficientes constantes, como se muestra a continuación:

$$V_i - k_b \times \omega(t) = R_i \times \frac{J \times \frac{d\omega(t)}{dt} + B \times \omega(t)}{k_p} + L_i \times \frac{J \times \frac{d\omega^2(t)}{dt} + B \times \frac{\omega(t)}{dt}}{k_p} \tag{B15}$$

De esta manera, la ecuación (B15) describe el modelo matemático para un motor de corriente continua, separadamente excitado. La obtención del modelo matemático se tomó del libro *Máquinas eléctricas*, de Fraile (2003).

Solución del modelo matemático del motor DC

El modelo matemático ya fue descrito y para su solución es necesario tener una consideración de mucha importancia: el valor de la constante L_i para motores de corriente continua separadamente excitado es aproximadamente cero, y siendo así la ecuación diferencial se transforma en una ecuación de primer orden, no homogénea, lineal y de coeficientes constantes:

$$V_i - k_b \times \omega(t) = R_i \times \frac{J \times \frac{d\omega(t)}{dt} + B \times \omega(t)}{k_p} \tag{B16}$$

Para el modelo se tiene como condición inicial que a un tiempo igual a cero (es decir, cuando el motor va arrancar) el valor de la velocidad es cero:

$$t = 0 \rightarrow \omega = 0 \tag{B17}$$

Así, ordenando, arreglando la ecuación (B16) y aplicando la transformada de Laplace a ambos miembros de la ecuación, se obtiene:

$$V_i = R_i \times \frac{J \times \frac{d\omega(t)}{dt} + B \times \omega(t)}{k_p} + k_b \times \omega(t) \tag{B18}$$

$$V_i = \left(\frac{R_i \times J}{k_p}\right) \times \frac{d\omega(t)}{dt} + \left(\frac{R_i \times B}{k_p} + k_b\right) \times \omega(t) \tag{B19}$$

$$\gamma = \left(\frac{R_i \times J}{k_p}\right) \quad \beta = \left(\frac{R_i \times B}{k_p} + k_b\right) \tag{B20}$$

$$\mathcal{L}[V_i] = \mathcal{L}\left[\gamma \times \frac{d\omega(t)}{dt} + \beta \times \omega(t)\right] \tag{B21}$$

$$\frac{V_i}{s} = \gamma \times s \times \omega(s) + \beta \times \omega(s) \tag{B22}$$

$$\frac{V_i}{s} = \omega(s) \times (\gamma s + \beta) \tag{B23}$$

$$\omega(s) = \frac{V_i}{s(\gamma s + \beta)} \tag{B24}$$

Una vez obtenida la ecuación de la velocidad en función del tiempo, se procede a resolver mediante fracciones parciales la ecuación (B24):

$$\omega(s) = \frac{V_i}{s(\gamma s + \beta)} = V_i \times \left(\frac{A}{s} + \frac{C}{\gamma s + \beta}\right) \tag{B25}$$

Los valores de *A* y *C* que satisfacen la ecuación son:

$$A = \frac{1}{\beta} \tag{B26}$$

$$C = -\frac{\gamma}{\beta} \tag{B27}$$

De esta forma, la ecuación queda descrita así:

$$\omega(s) = V_i \times \left(\frac{\frac{1}{\beta}}{s} - \frac{\frac{\gamma}{\beta}}{\gamma s + \beta} \right) \tag{B28}$$

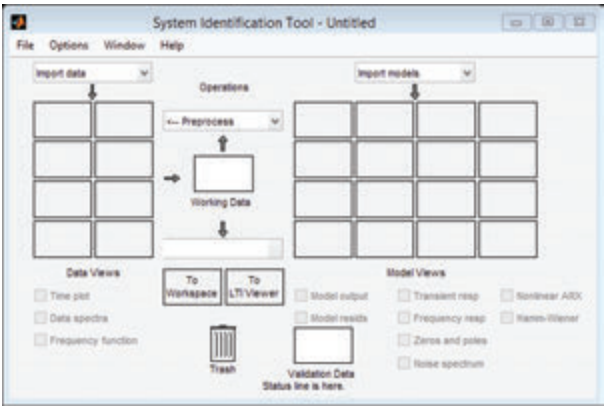
$$\omega(s) = V_i \times \frac{\frac{1}{\beta}}{s} - V_i \times \frac{\frac{\gamma}{\beta}}{\left(s + \frac{\beta}{\gamma}\right)} \tag{B29}$$

De la ecuación (B29) se observa que es una ecuación con 2 polos y 0 ceros.

Proceso para obtener la función de transferencia de un sistema

El proceso para obtener la función de transferencia de un sistema se observa en las figuras que se irán explicando a continuación.

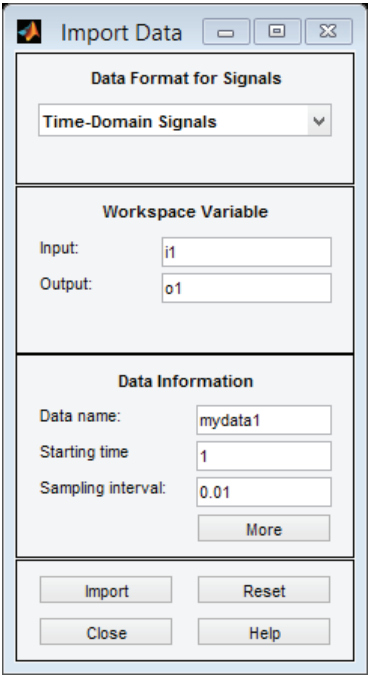
Figura 104
Ventana IDENT



Fuente: los autores

En la ventana de la figura 104 se deben ingresar los datos exportados del dominio del tiempo con un Sampling Interval de 0,01, ya que es el tiempo de muestreo propio de la placa Arduino Mega, como se muestra en la figura 105.

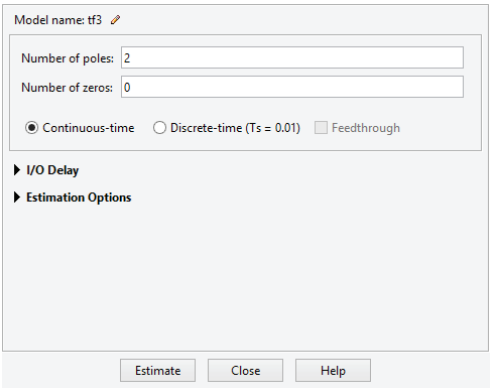
Figura 105
Ventana Import Data



Fuente: los autores

Con los datos importados se realiza un estimado de función de trasferencia para cada grupo de datos importados.

Figura 106
Número de polos y ceros del sistema



Fuente: los autores

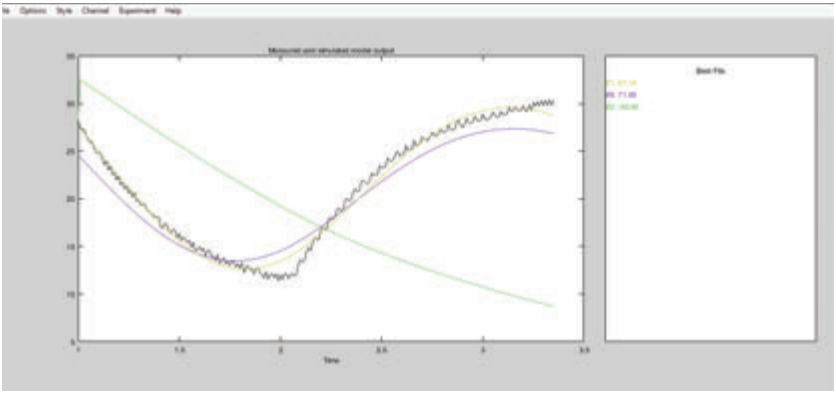
El siguiente paso es establecer el número de polos y ceros del sistema, como se visualiza en la figura 106, en la cual se eligieron 2 polos y 0 ceros, en base al modelo matemático del motor presentado en el apartado anterior. En la figura 107 se presenta la ventana de la herramienta IDENT de Matlab, en la cual se realiza la estimación de la función de transferencia.

Figura 107
Estimación de función de transferencia

Fuente: los autores

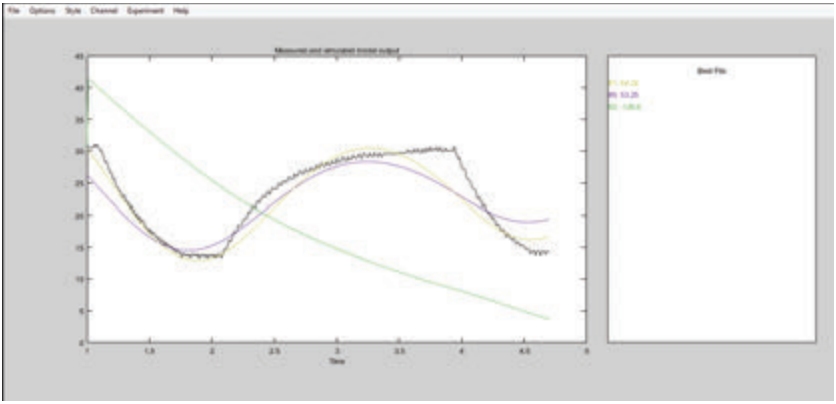
De todas las funciones de transferencia conseguidas se escoge la que presenta mejor aproximación, comparándolas respecto a las muestras importadas en la herramienta IDENT, con la ayuda de la opción "Model output", como se muestra en las figuras 108 y 109.

Figura 108
Modelamiento del sistema (primera muestra)



Fuente: los autores

Figura 109
Modelamiento del sistema (segunda muestra)



Fuente: los autores

De las funciones de transferencia calculadas, se determina la de mejor estimación, que en este caso es tf6, como se puede apreciar en la figura 110, donde se detalla la función de transferencia que mejor se adapta al sistema, presentando una aproximación de 85,91%.

Figura 110
Resultados presentados por Matlab

```
tf6 =

  From input "u1" to output "y1":
          0.5537
  ----------------------
  s^2 + 0.5606 s + 2.355

Name: tf6
Continuous-time identified transfer function.

Parameterization:
   Number of poles: 2   Number of zeros: 0
   Number of free coefficients: 3
   Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status:
Estimated using TFEST on time domain data "mydata3".
Fit to estimation data: 85.91% (stability enforced)
FPE: 0.7556, MSE: 0.73
```

Fuente: los autores

Bibliografía

Almeida, J., Silvestre, C. y Pascoal, A. M. (2015). Self-triggered state-feedback control of linear plants under bounded disturbances. *International Journal of Robust and Nonlinear Control*, *25*(8), 1230-1246. Recuperado de http://doi.org/10.1002/rnc.3138/

Antunes, D., Hespanha, J. P. y Silvestre, C. (2015). Stochastic networked control systems with dynamic protocols. *Asian Journal of Control*, *17*(1), 99-110.

Årzén, K. (1999). A simple event-based PID controller. *Proc. 14th IFAC World Congress*, *18*, 423-428. Recuperado de http://www.control.lth.se/documents/1999/arz99ifac.pdf/

Åström, K. J. (2008). Event based control. *Analysis and Design of Nonlinear Control Systems*, 127-147. Recuperado de http://doi.org/10.1007/978-3-540-74358-3_9/

Aström, K. J. y Bernhardsson, B. M. (2002). Comparison of Rienmann and Lebesque sampling for first order stochastic systems. *41st IEEE Conference on Decision and Control*, 2011-2016.

Åström, K. J., Hang, C. C. y Persson, P. (1989). Towards intelligent PID control. *Annual Review in Automatic Programming*, *15*(PART 2), 53-58. Recuperado de http://doi.org/10.1016/0066-4138(89)90010-4/

Bhuyan, S., Subudhi, B. y Ghosh, S. (2012). Networked Control System: Uncertainty Modeling and Stabilization. *Procedia Engineering*, *38*, 3662-3667. Recuperado de http://doi.org/http://dx.doi.org/10.1016/j.proeng.2012.06.422/

Bregulla Markus, Wolfgang Feucht, J. K. (2011). Wireless Solutions in Automation Overview and decision support. *ZVEI Automation*, *49*(1), 4-8.

Cervin, A. y Henningsson, T. (2008). Scheduling of event-triggered controllers on a shared network. *Conference on Decision and Control*, (2), 3601-3606. Recuperado de http://doi.org/10.1109/CDC.2008.4738939/

Chong, C., Kumar, S. P. y Member, S. (2003). Sensor Networks: Evolution, Opportunities, and Challenges, *IEEE*, *91*(8), 1247-1256. Recuperado de http://robertdick.org/sensor-nets/papers/chong-sensornets.pdf/

Cloosterman, M. B. G., Hetel, L., Wouw, N. Van De, Heemels, W. P. M. H., Daafouz, J. y Nijmeijer, H. (2010). Controller synthesis for networ-

ked control systems, *Automatica*, *46*(10), 1584-1594. Recuperado de http://doi.org/10.1016/j.automatica.2010.06.017/

Cominos, P., Munro, N., Lee, T. H., Ra, W. S., Jin, S. H., Yoon, T. S. y Park, J. B. (2002). PID controllers: recent tuning methods and design to specification. *Control Theory and Applications*, *149*(1). Recuperado de http://doi.org/10.1049/ip-cta/

Desborough, L. y Miller, R. (2002). Increasing customer value of industrial control performance monitoring-Honeywell's experience. *AIChE Symposium Series*, 169-189.

Donkers, M. C. F., Heemels, W. P. M. H., Van De Wouw, N. y Hetel, L. (2011). Stability analysis of networked control systems using a switched linear systems approach. *IEEE Transactions on Automatic Control*, *56*(9), 2101-2115. Recuperado de http://doi.org/10.1109/TAC.2011.2107631/

Donkers, M., Heemels, W. y Bernardini, D. (2012). Stability analysis of stochastic networked control systems. *2010 American Control Conference*, *48*(5), 3684-3689. Recuperado de http://doi.org/10.1016/j.automatica.2012.02.029/

Dorf R.C. (1962). Adaptive Sampling Frequency for Sampled-Data Control Systems;": *IEEE Transactions on Automatic Control*, *AC-7*, 38-47.

Dunbar, M. (2001). Plug-and-play sensors in wireless networks. *IEEE Instrumentation and Measurement Magazine*, *4*(1), 19-23. Recuperado de http://doi.org/10.1109/5289.911169/

Durand, S., Marchand, N., Durand, S., Marchand, N., Pid, A. E., With, C.,… Marchand, N. (2010). An Event-Based PID Controller With Low Computational Cost. *HAL Aechives-Ouvertes*.

Eberhart, R. y Kennedy, J. (1995). A new optimizer using particle swarm theory. *Sixth International Symposium on Micro Machine and Human Science*, 39-43. Recuperado de http://doi.org/10.1109/MHS.1995.494215/

ECMA-404. (1999). The JSON Data Interchange Standard. Recuperado de https://www.json.org/

Fernandez Barcell, M. (2008). Introducción a las redes de sensores inalámbricas. *Wireless Sensor Network*, 1-20.

Flores Carbajal, E. E. (2012). Redes de Sensores Inalámbricas Aplicado a la Medicina. *Universidad de Cantabria*, 1-15.

Fraile Mora, J. (2003). Maquinas Elécticas. España McGraw-Hill.

García, M. (2014). Análisis y diseño de una red inalámbrica de sensores para un proyecto agrario. *Universidad Oberta Catalunya*.

Guarnes, M., Dormido, S. y Visioli, A. (2009). Comparative Study of Event-Based Control Strategies: An Experimental Approach on a Simple Tank. *European Control Conference*, 1973-1978.

Guitiérrez, J. M. R. (2010). Implantación de Arduino en las redes Ethernet. *Arduinio Comunicación*, *1*, 3.

Heemels, W. P. M. H. y Donkers, M. C. F. (2013). Model-based periodic event-triggered control for linear systems. *Automatica*, *49*(3), 698-711. Recuperado de http://doi.org/http://dx.doi.org/10.1016/j.automatica.2012.11.025/

Heemels, W., Sandee, J. H. y Van Den Bosch, P. P. J. (2008). Analysis of event-driven controllers for linear systems. *International Journal of Control*, *81*(4), 571-590.

Hendricks, E., Jensen, M., Chevalier, a. y Vesterholm, T. (1994). Problems in event based engine control. *1994 American Control Conference - ACC '94*, *2*, 8-10. Recuperado de http://doi.org/10.1109/ACC.1994.752337/

Henningsson, T. y Cervin, A. (2009). Comparison of LTI and event-based control for a moving cart with quantized position measurements. *European Control Conf*, 3791-3796. Recuperado de http://www.control.lth.se/documents/2009/hen+cer_ecc09.pdf/

Hespanha, J. P., Naghshtabrizi, P. y Xu, Y. (2007). A survey of recent results in networked control systems. *95*(1), 138-172. Recuperado de http://doi.org/10.1109/JPROC.2006.887288/

Johansson, D. L. J. L. K. H. (2012). Comparison between sampled-data control, deadband control and model-based event-triggered control. *4th IFAC Conference on Analysis and Design of Hybrid Systems*, *7*, 7-12.

Johansson, M. y Jäntti, R. (2010). Wireless networking for control: Technologies and models. En *Networked Control Systems* (pp. 31-74). Springer Ed.

Kao, C.-Y. y Lincoln, B. (2004). Simple stability criteria for systems withtime-varying delays. *Automatica*, *40*(8), 1429-1434. Recuperado de http://doi.org/10.1016/j.automatica.2004.03.011/

Kwon, W. H., Kim, Y. H., Lee, S. J. y Paek, K. N. (1999). Event-Based modeling and control for the burnthrough point in sintering processes. *IEEE Transactions on Control Systems Technology*, *7*(1), 31-41. Recuperado de http://doi.org/10.1109/87.736747/

Laurent, D., Mitchell, J. y McDaniel, W. (1969). Adaptive sampling technique. *IEEE Transactions on Automatic Control*, *14*(2), 200-201.

Lehmann, D. y Lunze, J. (2011). Extension and experimental evaluation of an event-based state-feedback approach. *Control Engineering*

Practice, *19*(2), 101-112. Recuperado de http://doi.org/10.1016/j.conengprac.2010.10.003/

Lennartson, B. y Kristiansson, B. (2009). Evaluation and tuning of robust PID controllers. *IET Control Theory & Applications*, *3*(3), 294-302. Recuperado de http://doi.org/10.1049/iet-cta/

Li Yung, Kiam Heong Ang y Chong, G. C. Y. (2006). PID control system analysis and design. *IEEE Control Systems Magazine*, *26*(1), 32-41. Recuperado de http://doi.org/10.1109/MCS.2006.1580152/

Litz, L., Gabel, O. y Solihin, I. (2005). NCS-controllers for ambient intelligence networks-control performance versus control effort. *44th IEEE Conference on Decision and Control* (pp. 1571-1576).

Lizarazo, D. L. M. y Borja, J. A. T. (2015). Control de un Ball & Beam con matlab y lego nxt. *Visión Electrónica: Algo Más Que Un Estado Sólido*, *8*(2), 39-48.

Lunze, J. (2014). *Control Theory of Digitally Networked Systems.* (Springer, Ed.). Springer Cham Heidelberg New York Dordrecht London. Recuperado de http://doi.org/10.1007/978-3-319-01131-8_7/

Lunze, J. y Lehmann, D. (2010). A state-feedback approach to event-based control. *Automatica*, *46*(1), 211-215. Recuperado de http://doi.org/10.1016/j.automatica.2009.10.035/

Miskowicz, M. (2005). Sampling of Signals in Energy Domain. *International Conference on Emerging Technologies and Factory Automation*, *1*, 263-266. Recuperado de http://doi.org/10.1109/ETFA.2005.1612531/

Miskowicz, M. (2006). Send-On-Delta Concept: An Event-Based Data Reporting Strategy. *Sensors*, *6*(1), 49-63. Recuperado de http://doi.org/10.3390/s6010049/

Nilsson, J. (1998). Real-time control systems with delays. *Department of Automatic Control, Lund Institute of Technology (LTH).*

Oracle. (2018). MYSQL.COM. Recuperado de https://www.mysql.com/

Ortiz, F. (2005). Stability analysis of PD regulation for ball and beam system. *Conference on Control Applications, 2005. CCA 2005.*, 517-522. Recuperado de http://doi.org/10.1109/CCA.2005.1507178/

Otanez, P. G., Moyne, J. R. y Tilbury, D. M. (2002). Using deadbands to reduce communication in networked control systems. *American Control Conference*, *4*, 3015-3020. Recuperado de http://doi.org/10.1109/ACC.2002.1025251/

Pawlowski, A., Guzm, J. L., Rodr, F., Berenguel, M., Lenguajes, D. y Inform, D. (2008). Event-Based Control and Wireless Sensor Network for Greenhouse Diurnal Temperature Control : A Simulated Case Study. *Emer-*

ging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on, 500-507.

PHP. (2018). PHP 7.3.0 alpha. Recuperado de http://php.net/

Pi, R. (2017). The official Raspberry Pi Projects Book. Recuperado de https://www.raspberrypi.org/

Postoyan, R., Anta, A., Heemels, W. P. M. H., Tabuada, P. y Neˇ, D. (2013). Periodic event-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, *58*(4), 7397-7402.

Rabi, M. y Baras, J. S. (2007). Level-triggered control of a scalar linear system. *2007 Mediterranean Conference on Control and Automation, MED*, 1-6. Recuperado de http://doi.org/10.1109/MED.2007.4433671/

Rabi, M. y Johansson, K. H. (2009). Scheduling Packets packets for for Event-Triggered event-triggered control Scheduling Control. *10th European Control Conference (ECC)*, 3779-3784.

Rabi, M., Johansson, K. H. y Johansson, M. (2008). Optimal Stopping for Event-triggered sensing and actuation. *47th IEEE Conference on Decision and Control Cancun, Mexico*, 3607-3612.

Sánchez, J., Visioli, A. y Dormido, S. (2011). A two-degree-of-freedom PI controller based on events. *Journal of Process Control*, *21*(4), 639-651.

Sánchez, J., Visioli, A. y Dormido, S. (2012). PID Control in the Third Millennium. *Chapter Event-Based PID Control, Springer*, 495-526.

Santos, S. O. (2014). Sintonización de un controlador PID basado en un algoritmo heurítio para el control de un Ball and Beam. *Journal of Chemical Information and Modeling*, *53*(9), 1689-1699. Recuperado de http://doi.org/10.1017/CBO9781107415324.004/

Shi, Y. y Yu, B. (2009). Output Feedback Stabilization of Networked Control Systems With Random Delays Modeled by Markov Chains. *Automatic Control, IEEE Transactions on*, *54*(7), 1668-1674. Recuperado de http://doi.org/10.1109/TAC.2009.2020638/

Sridharan, G. (2002). Ball on Beam on Roller : A New Control Laboratory Device. *Conference on Industrial Electronics and Applications*, *91*, 1318-1321.

Tabuada, P. (2007). Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, *52*(9), 1680-1685. Recuperado de http://doi.org/10.1109/TAC.2007.904277/

Tarn, T.-J., Xit, N. y Bejczyt, A. K. (1996). Path-based Approach to Integrated Planning and Control for Robotic Systems*. *Automatica*, *96*(12), 1675-1687.

Vasyutynskyy, V. y Kabitzsch, K. (octubre, 2006). Implementation of Pid Controller With Send-on-Delta Sampling. *Ukacc*, 2-7.

Vasyutynskyy, V. y Kabitzsch, K. (2009). First order observers in event-based PID controls. *ETFA 2009 - 2009 IEEE Conference on Emerging Technologies and Factory Automation*, 1-8. Recuperado de http://doi.org/10.1109/ETFA.2009.5347082/

Walsh, G. C., Hong, Y. y Bushnell, L. G. (2002). Stability analysis of networked control systems. *IEEE Transactions on Control Systems Technology*, *10*(3), 438-446. Recuperado de http://doi.org/10.1109/87.998034/

Wang, X. y Lemmon, M. D. (2011). Event-eriggering in distributed networked control systems. *IEEE Transactions on Automatic Control.*, *56*(3), 586-601. Recuperado de http://doi.org/10.1109/tac.2010.2057951/

Wei, Z., Branicky, M. S. y Phillips, S. M. (2001). Stability of networked control systems. *Control Systems, IEEE*, *21*(1), 84-99. Recuperado de http://doi.org/10.1109/37.898794/

Weimer, J., Araújo, J. y Johansson, K. H. (2012). Distributed event-triggered estimation in networked systems. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, (Adhs 12), 178-185. Recuperado de http://doi.org/10.3182/20120606-3-NL-3011.00099/

Willig, A., Kubisch, M., Hoene, C. y Wolisz, A. (2002). Measurements of a wireless link in an industrial environment using an IEEE 802.11-compliant physical layer. *IEEE Transactions on Industrial Electronics*, *49*(6), 1265-1282. Recuperado de http://doi.org/10.1109/TIE.2002.804974/

Xu, K. A. J. (2015). *Advanced Discrete-Time Control Designs and Applications*. Singapur: Springer Ed.

Yajuan, C. y Qinghai, W. (2011). Design of PID controller based on PSO algorithm and FPGA. *2011 2nd International Conference on Intelligent Control and Information Processing*, *2*(2), 1102-1105. Recuperado de http://doi.org/10.1109/ICICIP.2011.6008424/

Yu, H. y Antsaklis, P. J. (2011). Event-triggered output feedback control for networked control systems using passivity: Time-varying network induced delays. *Conference on Decision and Control*, 205-210. Recuperado de http://doi.org/10.1109/CDC.2011.6160271/

Ziegler, J. G. y Nichols, N. B. (1995). Optimum settings for automatic controllers. *InTech*, *42*(6), 94-100. Recuperado de http://doi.org/10.1115/1.2899060/